**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Patent Application of:
Aymeric Perchant et al.

Application No.: 10/520,917 Confirmation No.: 5145

Filed: January 11, 2005 Art Unit: 2624

For: METHOD FOR PROCESSING AN IMAGE ACQUIRED BY MEANS OF A GUIDE CONSISTING OF A PLURALITY OF OPTICAL FIBRES Examiner: Andrae S. Allison

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**REVISED DECLARATION BY AYMERIC PERCHANT UNDER 37 C.F.R. § 1.131**

In connection with the Applicant's Response to the Final Office Action issued on April 7, 2009, this declaration sets forth the pertinent facts proving conception of the claimed invention prior to **December 2001**. Further, this declaration also sets forth the pertinent facts proving due diligence of the inventors from conception of the claimed invention until the filing date, July 18, 2002, of the priority application related to the present application (constructive reduction to practice).

I, Aymeric Perchant, hereby declare that:

1. I am a co-inventor of the subject matter described and claimed in the above-identified application, which relates to a method for processing an image acquired by means of a guide consisting of a plurality of optical fibres.

2. As evidenced by the attached copy of laboratory notebook pages and the corresponding English translations of the relevant notebook pages, we performed various testing of the claimed subject matter as early as July 2001, which is prior to December 6, 2001, the priority date of Shankar et al. (U.S. 6,885,801).

3. We had been diligent in working on this invention from prior to December 6, 2001 to the date of reducing it to practice. Evidence of diligence is attached [in French]. To accompany the diligence documents, the following spreadsheet contains an English description of each diligence document and how each diligence document relates to the image processing technology of the present application. The spreadsheet below also includes the date of each diligence document. The dates range from September 4, 2001 to November 4, 2002.

4. I hereby attest that the below English descriptions and dates corresponding to each of the diligence evidence documents are accurate and correct.

| Filename | Date | Title | Authors | Description/link with image processing |
|---|---|---|---|---|
| FMODIF_interface_cellViZio.ps | 18.Mar.2002 | Functional Requirement of Software Interface for Software prototype | Georges Le Goualher, Aymeric Perchant | Requirements of the first software prototype controlling the tomoscope, former name of Cellivizo prototype device, and including image processing capabilities. |
| MEMO_acquisition NOV2001.ps | 27.Nov.2001 | Image Acquisition in November 2001 | Georges Le Goualher | Image Acquisition protocol with the tomoscope prototype to acquire proper raw image for image processing testing. |
| MEMO_devlog.ps | 1.Feb.2002 | Software | Georges Le Goualher | Software architecture, including image processing. |
| Manuel/Utilisation/ | 5.Aug.2002 | Cellvizio User Guide | Georges Le Goualher, Aymeric Perchant | Software guide as a web page. Screenshots show the fiberdetect functionality, the background subtraction. |
| ReferenceImageCell-2002-02-07.ps | 7.Feb.2002 | Reference Guide of ImageCell module | Aymeric Perchant | ImageCell if the name of the image processing included in the Cellvizio (or tomoscope at the time) software. Drawings show the implementation of the fiber removal algorithm, and image reconstruction. Results from the Nov. 2001 database and algorithm block-diagram are shown. First release. |

| | | | | |
|---|---|---|---|---|
| ReferenceImageCell-2002-03-12.ps | 12.Mar.2002 | Second release of Reference Guide | Aymeric Perchant | Second Release. |
| ReferenceImageCell-2002-04-24.ps | 24.Apr.2002 | Third release of Reference Guide | Aymeric Perchant | Third Release. |
| Cahierdescharges PROTO.ps | 13.Sep.2001 | Functional requirements of the software prototype of the tomoscope | Georges Le Goualher, Aymeric Perchant | Includes image processing module and technical choices are addressed. |
| imageCellSA_cdc.ps | 4.Nov.2002 | Stand-alone interface for tmageCell | Aymeric Perchant | Command line specification for raw image post processing. |
| traitementImage.ps | 4.Sep.2001 | Image Processing | Aymeric Perchant | Optimal estimation of photon signal in the fiber on a raw image. Noise analysis. |

5. We filed an application in France on July 18, 2002, which is the priority application for PCT/FR03/02197. The present application is a national stage application based on this PCT application. The diligence evidenced attached shows that we had been diligent in pursuing the present invention from prior to Dec. 6, 2001 (the priority date of Shankar) to the date of filing of the PCT application, which is July 18, 2002.

6. All statements made of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Respectfully Submitted,

Date: April 16, 2009

Aymeric Perchant

Attachment.

Mauna Kea Technologies
9 rue d'Enghien, 75010 Paris

Ecole Nationale Supérieure des
Télécommunications
46 rue Barrault, 75634 Paris cedex 13

Rapport de Stage Ingénieur
Reconstruction d'images confocales acquises à travers un guide d'images

Sandra MARTI

Juillet-Décembre 2001

Engineering Internship Memoir

Reconstruction of confocal images acquired through an image guide

Sandra MARTI

July-December 2001

Results from Wednesday July 18th

- Number of pixels per fibers => OK
  - * histogram possible (program histo_fibres.m)
- imNbPix => image grey level of the fiber is its number of pixel.

- Number of neighbors per fibers => ok
  - * Display of percentage is possible

imNbVois => grey level in the image = the number of neighbors

Logical

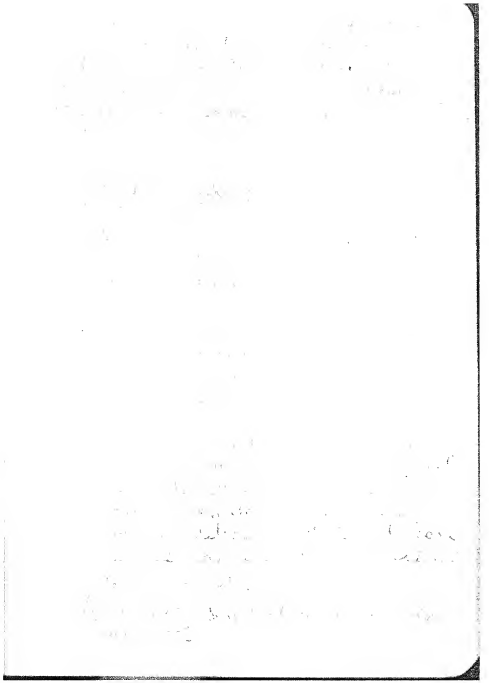
-Correlations:

The smallest fibers seem to have fewer neighbors.

The biggest fibers seem to have more neighbors.

Generally, there exists a trade off concerning the number of neighbors. Fewer are the isolated fibers that do not have 6 neighbors (not true on the borders).

Nonetheless, problems / abnormalities are not solely localized on the smallest fibers or the biggest.



Concerning the fibers the size of which are roughly normal, those abnormalities (number of neighbors not equal to 6) are localized near small blocks (that is, their isolation is sparse) => the structure is not regularly hexagonal (half regular)

Ideas for the future:

(1) look at the smallest fibers and their neighborhood. Use agreedy algorithm if needed

(2) Look at the biggest fibers and their neighborhoods.
Split them in two if necessary (Problem : how?)

(3) For the other ones, that have 5 or 7 neighbors, no fusion of split seems to be useful, as they have a normal size. We will never fall back to a regular hexagonal structure because this is not reality. () xxxxxx !

-> therefore, those fibers seems well localized.

02

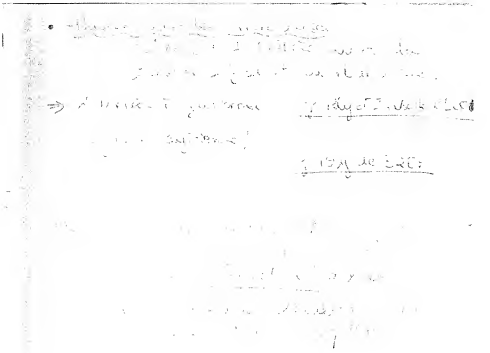
- July 19th,
  - Image pre-processing experiments

Ideas:
- median filter
- linear structuring element for closing (2 or 3 pix wide) in the 4 directions

Next Wednesday (July 25th)
-> code them
-> or in INRIA lib.

-> 4 closing (1 for each direction), then fuse them

-> how ?
(Decide later)



* display only the bad ones

-> Find one criteria linking the size of the fibers and its number of neighbors.

=> 2 ideas to merge cf. book of Isabelle Bloch

- bayesian fusion ?
- cf. book of BREF.

NB: results for the detection of the fibers are better when we first begin with a zoom, then a closing before.

Therefore, best results on imcrop than on imCropVrai

Back to the theoretical calculation of the number of pixels per fiber

* Apply to the image: imCrop

* Size = 490 * 480 = 235200

* n_fibers= 4278

Surface = 4278* 13.86 10^-12 = 5.929.10^-6 m^2

x= total nb pixel total / nb of fibers = 54.98

x= 55 pixel / fiber

---

July 19th

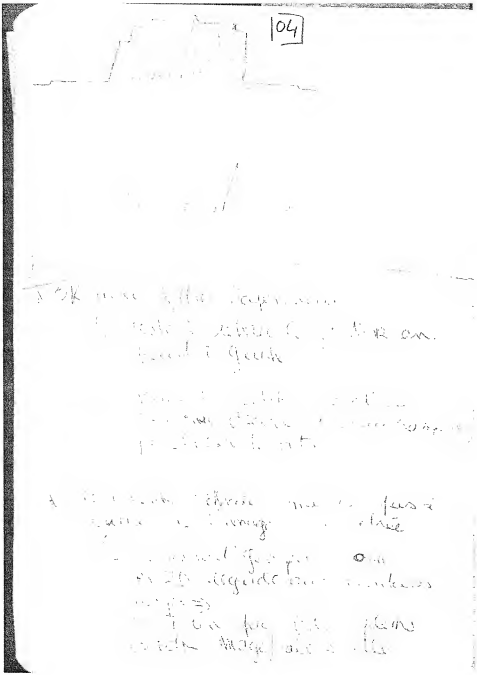
* Set to zero the contour (external part of the bundle)

* 640 pix

640 pix

Method = watershed (LPE = Ligne de Partage des Eaux) with markers / fiducials

cf. p 135 of Matioli.



July 20th, evening

* OK with mean filtering

-> still one black dot to remove on the upper left

-> Make a small dilation or erosion (depending on the mask level) then, smooth the contour.

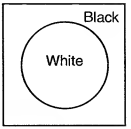
* Previous method does not work because of the input image

-> Geodesic reconstruction by erosion in 2D works on the neighborhood of the pixels =>

If there exists one isolated pixel (plenty in our images) then it[...]

will leak on the borders => not good.
So we give up that.

* Then, thresholding to obtain the background and the image guide, such as:




= mask

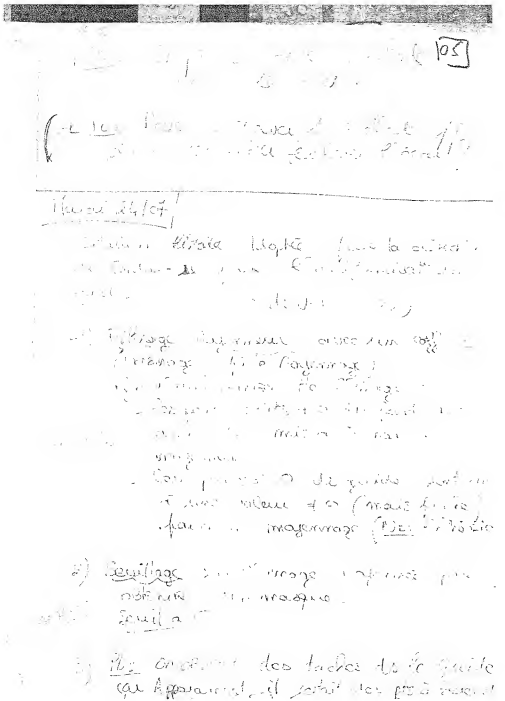
July 23rd, monday

With mean filter:

There exists problems, because the choosen coefficient (here 3 = neighborhood of the averaging) is not good for all the images.

Either we still have points different from zero in the background, or we have points equal to zero in the bundle / guide, thus the probems for the thresholding!

Furthermore, we are not sure that the masking stick to the bundle / guide (because the mean filtering distorts the contours).



NB: moreover, this method smoothes the contour.

==> Pb: Still to be found is a method to calibrate it correctly (remove the offset ?

---

**Tuesday July 24th**

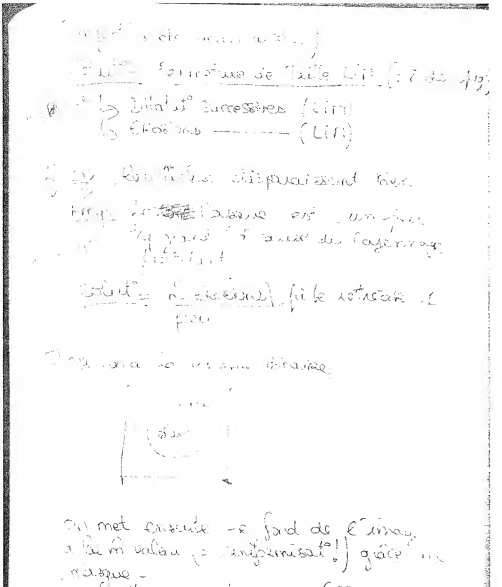
Solution adopted to detect the contour,
i.e. to make the background uniform:

==> in Version 00

1) Average filtering with a cooefficient = 3
   (neighbourhood filtering, for averaging)
   to make the whole image uniform:
   - Small dots ≠0 will be set to 0 with this averaging
   - Small dots = 0 within the guide are set to a value ≠0 (but close to) with the same averaging (NB : in theory)

2) thresholding of the averaged image to get a mask
   threshold set to 0

3) Pb : some spots exist within the guide because, apparently, some points with zero value were remaining, where



the injection was bad ...)

Solution : LIM mesh closure (=8 in the code)

=> Successive expansions (LIM)

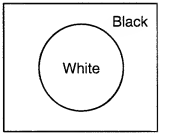
=> Successive erosions (LIM)

4) Ok. Spots disappear.

Remark: The mask is a bit too large (because of the preceeding evaraging)

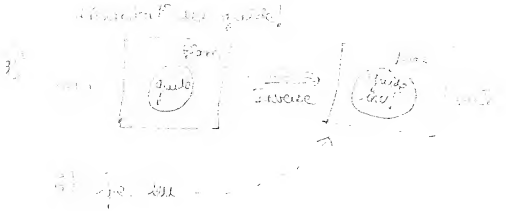
Solution: 2 erosions to make it slightly smaller

5) Ok. We have a binary mask.

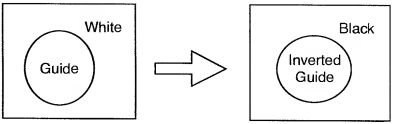



Then the image background is set to the same value (= unformity) thanks to the mask.

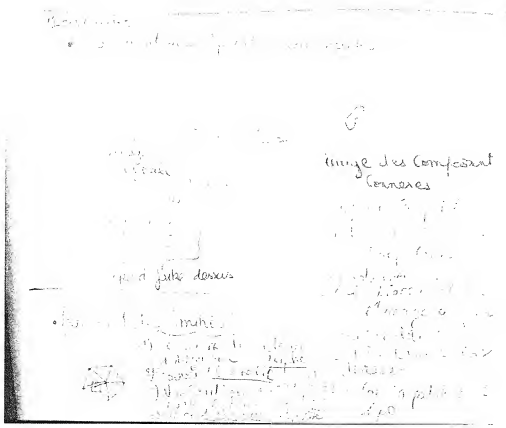
One choses to set it to 255 (white), so that when the image is inverted to perform watershed, the background will be seen as a large basin, which will allow to get a good



delineation of the guide !




6) watershed over



For the next steps:

Processing of the « patatoes-peanuts »


watershed to be done on it

1) to locate the "patatoe"

2) remember its connected element number

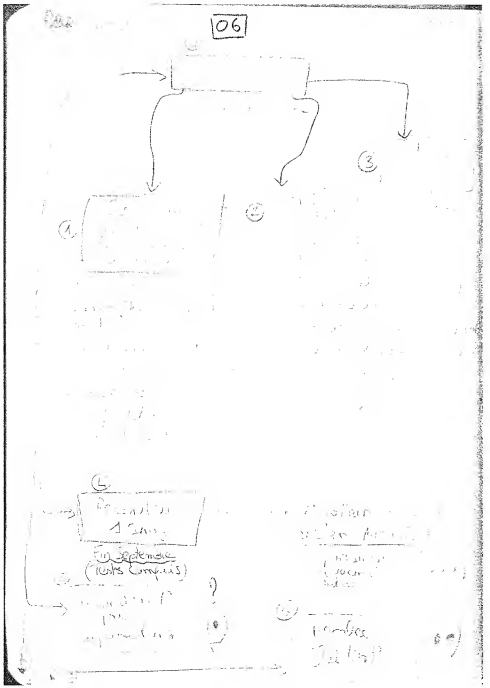
3) relocate it on the original image

4) check in the original image that it contains 2 maxima

5) isolate the "patatoe", then watershed

Processing of the "mini (fibers)"

1) make use of the adjancency table
=> graph
2) find a good library
3) see a solution which improves the environment
=> with a counter (+1 if nb of connected elements is not 6)




INTERNSHIP: July. 24th:

Image

0 Fiber

Isolate each fiber
Get the good ones

3 Control of the guide

(**)

1 Estimation of the flux backscattered by the tissue for each fiber

2 Fiber tracking

Statistically, each fiber is hit several times.
Not only find the good spot => for statistics,
Use all the spots of one fiber to retrieve the "true" value of the fiber

Ideally, the guide is tangent (almost) to the edges.
No amplitude variation of the sampling window

(*)

4 Reconstruction of 1 image

End of September
(including tests)

Still to know: has the pattern to be removed or not (blind tests)
- Image experts
- Physicians
- Others

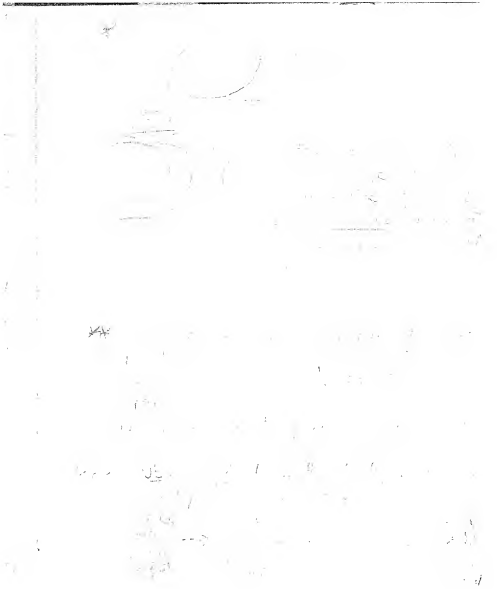
(***)

5 Reconstruction through super-resolution (.)

6 Re-positionning Stabilization (..)

7

8



(*)

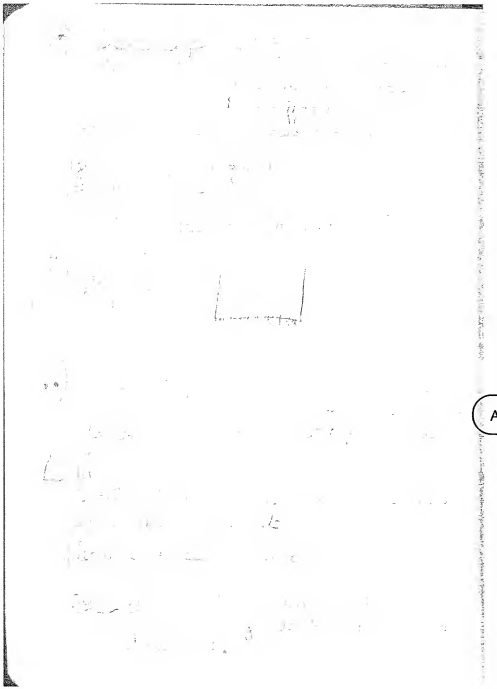
Theory

Shift of the guide within the image, which we will be able to fix if we know how to re-position it, thanks to the knowledge of the center of the fibers

(**) In theory, the manufacturer guarantees that a single fiber cannot break. If the guide breaks out, it breaks entirely. But it is worth knowing how to check it by ourselves.

(***) NB: we may imagine a degree of visibility of the fiber pattern.

Image with complete pattern ⋯→ Image with weak pattern ⋯→ Image with no pattern at all



(•) In case of small movements (< core-to-core distance)
Super-resolution = resolution better than the intrinsic resolution of the instrument

obtain a better sampling

the grid is not regular
fiber network

the movement is randomized (unknown)

but we don't know whether super-resolution will be possible

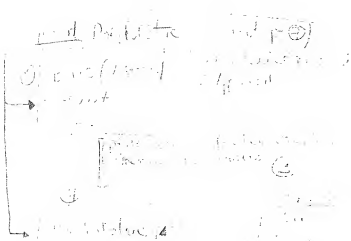
(••) we don't know how to re-position and stabilize images if the fiber pattern is there.
Re-positionning, as if there has not been any fiber.

A

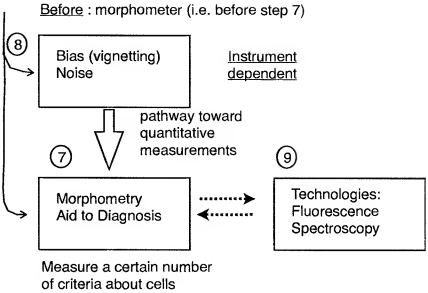
HF (high frequency) = instrument
If some HF exist in the images, it must really correspond to objects.

3D re-positionning => volume

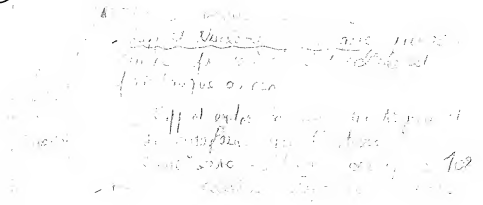
Stabilisation = keep an image stable when we observe the image sequence. Find a compensation



because, if not, we just re-position the pattern, and we do need this only to check the guide.




A



To include in the memoir

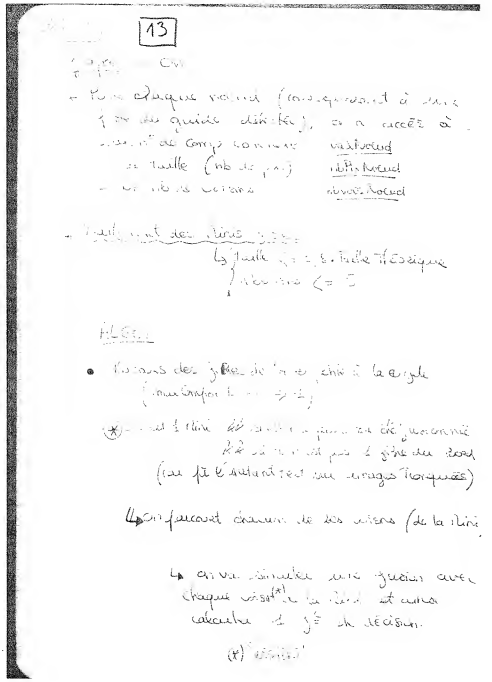
Any morphologic measurements

- Nucleo-cytoplasmic ratio, which is the criterion to decide whether a cell is pathologic or not

  => Ratio beween the size of the nucleus and the cytoplasm which surrounds it.

  healthy cell => nucleus takes 10%

- Cell shape (polygonal, or circular)



August 31st.

Graph => OK

* For each node (corresponding to one detected fiber of the image guide), we have access to:
- its index of connected component valNode
- its size nbPixNode
- its number of neighbors nbNeighNode

* Processing of the tiny fibers

-> size <= 0.8 * theoretical size
nb neighbors <= 5.

Algorithm:

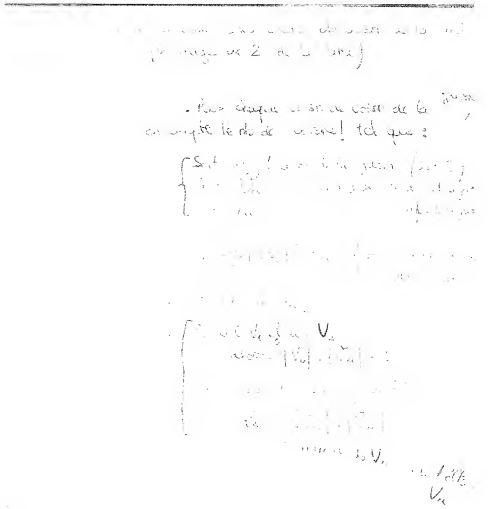
* Scan the fibers from the smallest to the biggest
* (Maxcomponent + 1 -------> 1)

* If it is a small / mini fiber and if it has not been merged and it is not on the border
(because for the moment we test on cropped images)

-> wwe scan each neighbor (of the tiny / small)

-> we simulate the merge with each ,neighbor (*) of the tini/small and compute one decision function / criteria

(*) neigh mini



(1) - we count all the nieghbors of te neighbor of the mini
(2-neighborhood of the mini)

* for each neighbor of the neighbor of the merge, we count the number of neighbors! such as:

  Let u be a neighbor of the merge (v1 + v2)
  Let Vu be the neighborhood of u before the nerge
  Let Vu be the neighborhhod of u after the merge

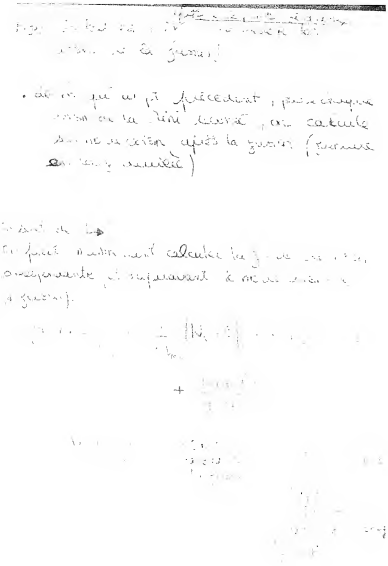
* V1+2 = (V1 U V2) \ {v1,v2} = neighborhood of the merge (v1+v2)

* u belongs therefore to V1+2
* If u belongs to V1 and u belongs to V2
  then |Vul=|Vul-1
  Else, i.e. u belongs to V1 or u belongs to V2

  then |Vul=|Vul

  = Cardinality of Vu = Number of elements of Vu

(2) - we count all the neighbors of the mini
(all but the neighbor "neigh mini" with which we test the merge and the nodes already counted at the previous step in the 2-neighborhood of the mini)



NB: the aim here is, after those 2 points (1) and (2) to have counted all the neighbors of the merge!

As in the previous point, for each neighbor of the current mini, we compute its number of neighbors after the merge (formula in red below)

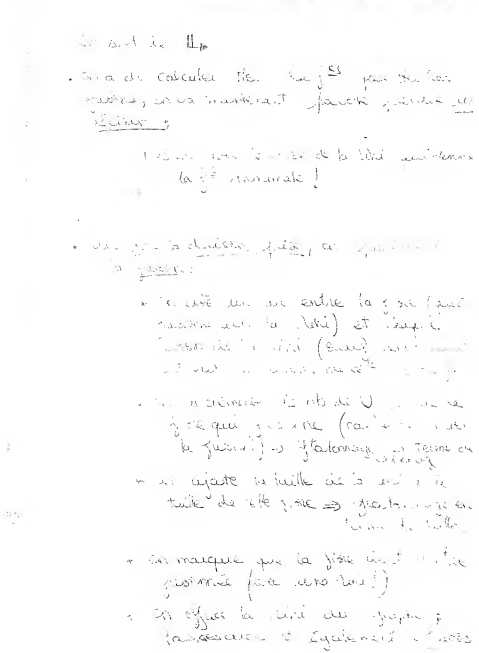
We get out of ->
We can now compute the corresponding decision function / criteria (and before, the number of neighbors of the merge)

$$function = \frac{1}{N_{m+i}} \sum_{v\ in\ V_{m+i}}^{\parallel} |N_v\text{-}6| + \alpha |N_{m+i}\text{-}6| + \frac{merge\ size}{theoretical\ size}$$

Number of neighbors for the merge

Over all the neighbors of the merge

In order to give a higher priority on the smallest fiber of the neighborhood of the merge.



We get out of ->

* Therefore, we have computed all the functions for all the merges, we are now going to <u>decide</u> :

Merge with the neighbor of the mini that have the smallest function

Once the decision is taken, the merge is computed :

* we create an arc betwee, the fiber (the one that merges with the mini) and each neighbor of the mini but the one that are allready neighbors of this fiber)..

* we increment the number of neighbors of the fiber that merges (because "it becomes the merge") => greedy in terms of neighborhood.

* we add the size of the mini to the size of the fiber => greedy in terms of size

* we label the fiber that has just been merged (with a mini)

* we remove the mini from the graph ; all arcs are removed too.

* we keep track of the merge in a table (vector) with 2 dimensions.

We get out of (*).

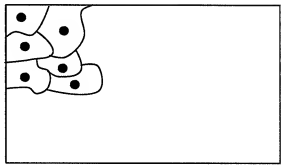
We get Out of .

Sept 3rd.

Tab

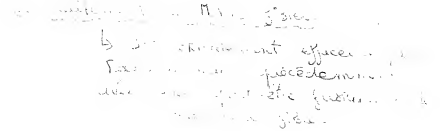
| Mini | Merged Nighbor |
|---|---|
| a | O |
| A | B |

Image of connected components



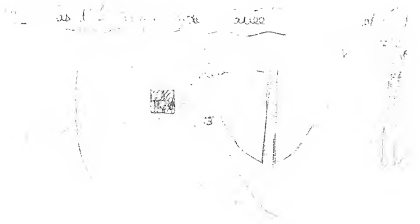
Processing of the big fibers: ==> Ok

=> sometimes creates very small fibers, like this :

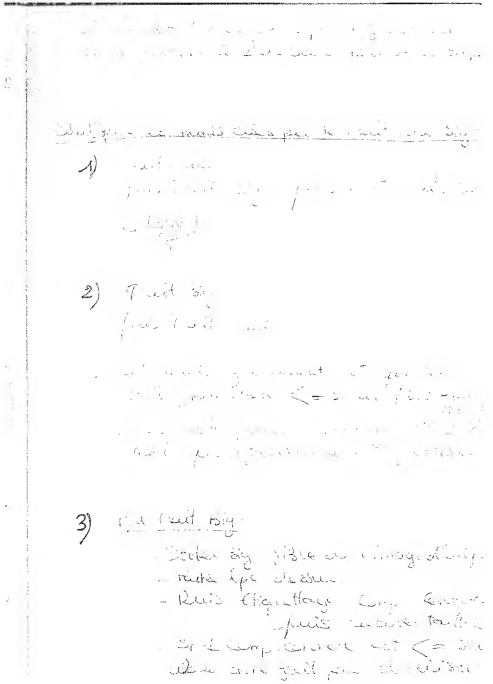


Processing of the mini fibers:

=> should normally fix this sort of problem, but the minifiber previously created will perhaps be fused with a wrong fiber



Pb: situation of a "square" mini-fiber



If one removes one of its pixels to fuse it, one fuses it to its two neighbours in the same time

Solutions for the mini-fibers created by the processing of big ones

1) Processing for mini-fibers, then processing for big fibers, athen, again, processing for the mini-fibers

==> long !

2) Processing for big fibers, athen, processing for the mini-fibers

!!! if 2 mini-fibers fuse together and if teh resulting size is below the threshold (0.8 * theoretical size), then the resulting fusion will again be considered as mini-fiber and will have to be fuse too.

3) during the processing of the big fibers

- isolate teh big fiber from the original image
- process watershed on it
- labelling of connected elements, then computing the size
- if one connected element is < threshold, then no division...

... of the big fiber (we leave it as it is).

==> add the compacity criterion (S/p^2), maximum for a circle

LIST of the functions used for detecting and removing the fiber pattern

Mathematic morphology

Numeric closure
=> to enhance the areas between fibers

Binary closure (size > 1)
Binary dilatation => N
Binary erosion => N

To close holes while the mask is created

Geodesic reconstruction through numeric erosion

While creating the mask, in order to erase the possible "seams" (of the mask on the edge of the guide) which may have occured during binary closure of sise N

Geodesic reconstruction through numeric dilatation

During h-dome method to locate local maximas of one fiber

Averaging filter

to create a fuzzy image which will be used to build the mask of the image guide

Add or Remove one border to the image

and set a value to this border

Invertion (of the grey levels) of an image

to apply the watershed algorithm on it

Watershed algorithm

because the injection profile on the guide is similar to the egg-box model.

Cf Morpho Maths

Valleys = between fibers;
Summits = maximum of each fiber

Labeling in connected elements

after watershed, in order to give each fiber a number and locate it within the image

Shift functions to wander across an image

points
rows
slices

"offset box" = allows to go across the neighbour pixels in any connectivity

SetImageLevel: set the whole image to a constant level

Comparison functions between 2 images

MIN, MAX,...
SUM, SUB,...

Image uniform modification functions

SUB_c
SUM_c, ...

**Diligence Document 1**

# MODIFICATION DE L'INTERFACE GRAPHIQUE DU MODULE D'ACQUISITION CELL-VIZIO

Auteur du mémo : Florence Meusburger

Date : 11 03 2002

**Diffusion :** interne

**Section :** développement informatique

**Version :** *Revision* : 1.5

**Référence du document :** `$Id: FMODIF_interface_cellViZio.tex,v 1.5 2002/03/18 12:51:19 florencem Exp $`

## 1 Introduction

Ce document est le cahier des charges des modifications à apporter à l'interface graphique actuelle du prototype Cell-viZio, dans le cadre de la mise en conformité du prototype avec la règlementation applicable (APAVE).

## 2 Modifications :

**Lancement et sauvegarde de l'acquisition d'images :**
Seuls deux boutons permettent de commander l'acquisition des images : le bouton *Start/Stop* et le bouton *Save*. On supprime le bouton *Pause/Continue* qui n'a pas grand intérêt pour l'utilisateur standard.

**Langue :** Il faudrait que l'interface soit disponible dans diverses langues : l'utilisateur choisirait celle souhaitée. Pour l'instant, l'anglais est imposé.

**Paramètres électroniques :**
Les paramètres électroniques une fois réglés n'ont plus besoin d'être modifiés. C'est pourquoi, la zone de saisie des paramètres électroniques disparait de la fenêtre principale du module d'acquisition. Mais, il faut pouvoir modifier ces paramètres en cas de déréglage. L'entrée "Parameters" du menu "Calibration" entraine l'ouverture d'une fenêtre de modification des paramètres électroniques. L'accès à cette fenêtre doit être limité aux personnes autorisées. Pour cela, deux possibilités :

- On peut mettre en place un système de login au lancement de l'application permettant de différencier les utilisateurs. On distingue deux "types" d'utilisateurs de l'application : le "simple utilisateur" et "l'administrateur". Seuls les "administrateurs" ont accès à la fenêtre des paramètres. Ceci implique qu'un utilisateur se charge de créer un "compte utilisateur" pour chaque utilisateur, et que chaque utilisateur se logue pour pouvoir lancer l'application, ce qui est tout de même contraignant. Cependant, cela permettrait de garder la trace dans un fichier des différentes actions des utilisateurs, mais est-ce vraiment utile ?

- Une autre solution consiste à demander un mot de passe à l'ouverture de la fenêtre des paramètres. Seuls les détenteurs du mot de passe ont accès à la fenêtre des paramètres. Ce mot de passe est entré une seule fois, à la première ouverture de la fenêtre.

La fenêtre des paramètres ressemble fortement à la zone des paramètres actuelle. Le bouton *send* est supprimé (pour éviter d'oublier...) : tout changement entraîne la modification immédiate de l'électronique. L'utilisateur a la possibilité d'enregistrer une configuration de paramètres avec le bouton *save*, configuration qui sera automatiquement chargée au lancement de l'application. Suite à des modifications des paramètres, il peut recharger la configuration sauvegardée avec le bouton *load*. Cette fenêtre peut rester ouverte pendant l'acquisition.

Certains paramètres électroniques sont cependant directement accessibles dans la fenêtre principale du module d'acquisition et modifiables par tout utilisateur : la résolution de l'image (640x640, 512x512, ou 384x384), et le "horizontal time shift". Leur modification est prise en compte instantanément.

**Traitements d'images :**
La fenêtre de traitement d'images est conservée : l'utilisateur a le choix du nombre d'images à "bufferizer", il peut lancer l'acquisition de fond avec le bouton *Background*, soustraire le fond avec le bouton *Subtraction*, et lancer le traitement d'une image avec *ImageCell*. En revanche, l'évolution des boutons *FiberDetect* et *TempMean* va dépendre de la faisabilité de traitements en parallèle de l'acquisition.
Il faudrait mettre en place un système qui, au cours de l'acquisition, détermine si la détection des fibres estimée auparavant est correcte, ou si elle doit être refaite. Ainsi, on détecte les fibres au lancement de l'application, puis quand le système le juge nécessaire, suite à l'accord de l'utilisateur. Le bouton *FiberDetect* sera alors supprimé.
Pour la moyenne temporelle, l'idéal serait de la calculer régulièrement et de la soustraire aux images acquises ; il reste à déterminer si on a le temps de le faire en temps réel... Si tel est le cas, le bouton *TempMean* deviendra un "toggle button" : la moyenne sera soustraite si et seulement si le bouton est enfoncé.
Ainsi l'interface ne contiendra plus que quatre boutons de traitement : *Background*, *Subtraction*, *TempMean* et *ImageCell*.

**Sécurité laser :**
Lorsqu'on démarre le soft, le laser est automatiquement éteint. L'allumage du laser s'effectue deux secondes après le lancement de l'acquisition avec *start*, pendant lesquelles un message clignotant "laser on" apparaît à l'écran. Pendant ces deux secondes, la procédure d'activation de l'émission laser peut être interrompue par un *stop*. Puis un témoin bleu bien visible à l'écran indique que le laser est allumé, pendant toute la durée de l'émission. L'appui sur le bouton *stop* entraîne l'arrêt du laser.

**Sauvegarde du fonctionnement du logiciel :**
Il serait intéressant d'enregistrer dans un fichier quand et pendant combien de temps le logiciel a été utilisé, ainsi que les exceptions générées au cours de l'utilisation du logiciel ; ceci afin d'estimer le bon fonctionnement du logiciel et de déceler les bugs.

**Améliorations à apporter au viewer :**
Voici une liste non exhaustive des améliorations envisageables pour le viewer :
- trouver un moyen de diminuer le scintillement engendré par la visualisation d'un film.
- dans la ViewPalette, les seuils de la palette de couleur ne sont pas manipulables de manière intuitive. Pour l'instant, on fixe les seuils à l'aide de clics droit ou gauche de souris. Il serait

peut-être plus facile d'ajouter des "barres" sur la ViewPalette, matérialisant les seuils.
- On n'analyse pas l'image ou le film affiché afin de trouver sa palette de couleur. Quand on affiche la palette associée à l'image ou au film ouvert, on suppose toujours que c'est une palette en niveaux de gris...
- permettre à l'utilisateur de choisir la zone à zoomer à l'aide de la souris. Pour l'instant, la zone zoomée est la région en haut à gauche de l'image. On peut par la suite se déplacer dans l'image zoomée à l'aide de la souris ou des flèches.

**Diligence Document 2**

# ACQUISITIONS D'IMAGES NOVEMBRE 2001

Auteur du mémo : Georges LE GOUALHER

Date : 27 Novembre 2001

**Diffusion :** interne, NA
**Section :** optique-image
**Sujet :** constitution d'une base de données images
**Version :** *Revision* : 1.3
**Référence du document :** `$Id: MEMO_acquisitionNOV2001.tex,v 1.3 2001/11/27 16:22:38 georges Exp $`

## 1 Objectif

Faire une série d'acquisitions afin de constituer une base de données d'images significatives montrant le savoir faire de MKT en acquisition d'images de rétrodiffusion en novembre 2001.

Ces images feront l'objet d'un mémo qui présentera les images acquises en mode non-fibré, en mode fibré et des traitements d'images qui leur seront appliqués pour montrer l'apport du traitement de l'image. On veut donc acquérir des images comparables en mode fibré et non-fibré (amplitude spatiale, grandissements similaires).

## 2 Protocole

On désire faire des images en mode fibré et non-fibré dans les meilleures conditions possibles.
Pour le mode fibré, on utilise donc la tête dépliée et différents grandissements (i.e. différents objectifs :

- x25 (Gtête=1 ; Gsystème=6) ;
- x40 (Gtête=1.6 ; Gsystème=9.6) ;
- x100 (Gtête=4 ; Gsystème=24) ;

Pour le mode non fibré, on utilise un objectif :
- x25 (Gsystème=6) ;
- x40 (Gsystème=10) ;
- x100 (Gsystème=25) ;

Pour chacun des éléments, on enregistre des images à différentes profondeurs. Le pas en Z peut être adapté en fonction de la résolution axiale au grandissement courant.
Les éléments à imager sont :
- la mire USAF ;
- la mire de distorsion ;

- le circuit intégré ;
- l'oignon ;
- muscle de poulet ;
- des tissus (à définir) ;

Points importants lors de l'acquisition :
- Pour chacunes des acquisitions il faut connaître l'amplitude du champ balayé (afin de déterminer la résolution de l'image acquise) ;
- Il faut être capable de connaître la résolution latérale et axiale des images ;
- La profondeur doit être connue ;
- Une acquisition du fond (réflexion parasite) est enregistrée (20 images) ;
- Les images ne doivent pas être *trop* saturées ;

## 3 Organisation des données

/Nov01/fibre/USAFGsys1, USAFGsys1_6,..., Distorsion, Oignon, Tissus(à préciser)
/Nov01/nonfibre/USAFGsys1, USAFGsys1_6,..., Distorsion, Oignon, Tissus(à préciser)

**Diligence Document 3**

# DÉVELOPPEMENTS LOGICIELS

Auteur du mémo : GLG

Date : 02 01 2002

**Diffusion :**
**Section :**
**Sujet :**
**Version :** *Revision* : 1.2
**Référence du document :** `$Id: MEMO_devlog.tex,v 1.2 2002/01/02 18:15:01 georges Exp $`

## 1 Introduction

Le but de ce document est de définir les développements logiciels à effectuer sur l'application principale du tomoscope **mktApp**. Le texte en italique réprésente des éléments non prioritaires. Les références aux M1, M2 etc... font référence au CR de Réunion Image avec le E. Borotto.

## 2 Application Tomoscope : mktApp

### 2.1 Creation du Module de Visualisation : ModVisu

Le module de visualisation devrait être un composant Qt utilisable en tant que tel, utilisée dans le module d'acquisition **modAcq** (zone d'affichage) et dans le module de post-traitement **modProcessing**.

Le module de visualisation permet de visualiser une image fixe (**8 ou 10 bits ou plus**) de type BMP, PNG ou un film INR ou MNG (**8 ou 10 bits ou plus**) sous forme de séquence animée.

Opérations pour la visualisation ([1]) :
- le module de visualisation permet de visualiser une image fixe (**8 ou 10 bits ou plus**) de type BMP, PNG ou un film INR ou MNG (**8 ou 10 bits ou plus**) sous forme de séquence animée.
- effectuer un zoom (avec les boutons + et -)
- translater l'image (avec les flèches)
- appliquer une LUT (niveau de gris ou fausses couleurs). L'ensemble des LUTs pourra être évolutif (ajout facile de nouvelles LUTs). L'utilisateur peut agir sur les niveaux min et max pour faire un fenêtrage (linéaire ou logarithmique) . On peut également inverser les niveaux de gris.
- appliquer une transformation gamma sur le fenêtrage courant.
- afficher l'information associée à l'image (**M5**) (commentaire contenu dans le fichier `cfg` si celui ci est disponible). L'utilisateur doit pouvoir modifier le commentaire associée (est-ce que tout les utilisateurs peuvent le faire ?). Cela permet à l'utilisateur de compléter et/ou rectifier l'information associée à une image.

[1] cf. display.exe

- Faire un outil interactif permettant de tracer un trait entre deux points et d'afficher la longueur de ce trait.
- *possibilité de sélectionner une zone pour y faire une opération (mesure de surface, histagramme)*
- *afficher des règles sur les cotés (indices des pixels ou coordonnées micrométriques)*
- *afficher les axes indiquant la position du curseur*
- *inversion des niveaux de gris*

*Opérations pour l'export :*
- *exporter (enregistrer) l'image courante (avec application de traitements éventuels) dans un format adéquat avec marquage MKT optionnel en tant que tel ou dans un traitement de texte en vue de l'édition d'un rapport.*

# 3 Modifications pour le module d'acquisition : modAcq

## 3.1 Modifications identifiées

- Cf. Sourceforge
- RealTomoscopePara : singleton
- M1
- M3
- M10

## 3.2 Modifications à approfondir

- M11 ? (Remarque : influence sur le module de visualisation)
- **Intégration du module de visualisation** : La zone d'affichage doit être remplacé par le module de visualisation.
- **Optimisation de la procédure d'acquisition et du traitement associé** : (méthode doIt() de AcquisitionThread) : il faudrait optimiser les accès aux données et calculs effectués. Un flowchart (parcours des données, calculs) précis devrait être représenté afin d'optimisé la boucle d'acquisition. Pour effectuer cette modification, il faudra prendre en compte les besoins en TDI (cf. section 6).
- **Choix de l'interface d'acquisition :** On doit pouvoir passer rapidemment d'un type d'acquisition vers un autre : mode simulé, mode AMCC, mode NI1422, mode FireWire. Cela peut être fait à partir d'une classe générique (interface) et de différentes implémentations dérivée. La classe générique est à définir (elle peut par exemple contenir les méthodes open(), close(), reset(), getData(),...) -voir dans le cas de la carte NI et du FireWire si cela est compatible-. On n'utilisera que les fonctions minimales de la carte d'acquisition (getData()), les traitements (application de LUTs, inversion des lignes) seraient fait en soft. Le point dur peut être de faire en sorte que le passage 8 vers 10 bits ou plus soit quasi transparent.
- **Fonctionnement en mode 10 bits :** Ce problème est lié à la tâche précédente. En bref, il s'agit d'étudier les points suivants : Acquisition des données en mode 10 bits, Prise en main de la carte NI ; Format d'enregistrements des données ; Affichage des données (choix d'une LUT, d'un gamma) ; Distinction des données brutes avec les données affichées et/ou traitées.

# 4 Modifications pour le module browser : modBrowser

Les modifications de ce module sont liées à la tâche de définition d'une base de données d'images (cf. ...) .
Parmis les points déjà évoqués :

- **Sauvegarde des données :** il faut un utilitaire permettant de faire une sauvegarde aisée des images sur CDs. Il faut pouvoir extraire des paquets de 500 Mo correspondants à des images acquises à partir d'une date donnée.
- **Droits d'accès aux données.** On doit pouvoir visualiser les images à partir de différents postes de travail (**mktApp** sans le module d'acquisition **modAcq**) , cependant il faut distinguer la personne qui fait l'acquisition des images (opérateur) qui doit pouvoir effacer des images, les renommer,... des personnes qui consultent les images qui n'ont pas de droit.
- **Prévisualisation** : aujourd'hui dans le module browser toutes images ont la même vignette. On pourrait associer une vignette de prévisualisation à chacune des images (est-ce pertinent ?)
- **Recherche rapide d'images** On voudrait retrouver de façon rapide des images à partir de mots-clés. Ces mot-clés peuvent se raporter au nom du fichier image et/ou aux informations associées aux images (présentent dans le fihier .cfg)

# 5 Modifications pour l'application : mktApp

- Problème de l'affichage module d'acquisition/browser/visualisation. Lorsque l'on créé un nouveau module d'acquisition depuis le module browser, celui-ci n'apparait pas en premier plan.
- Doit-on mettre un "setup" permettant de choisir la carte d'acquisition, l'affichage de certains éléments (mode expert ou mode utilisateur par exemple).
- Assistance au protocole d'acquisition : peut-on intégrer un controle actif d'un protocole dans l'application ?
- Utilisation de XML pour gérer l'ensemble des paramètres ?
- Veille complémentaire : il faudrait regarder des applications similaires à celle que l'on doit développer pour s'en inspirer éventuellement. Qu'en est-il du web ? De l'insertion dans un milieu médical ?

# 6 Traitement de l'image : TDI

Les traitements d'images identifiés sont :

## 6.1 TTRQTR

- **soustraction temps réel du fond lié à la réflexion parasite**. Le module d'acquisition permet d'enregistrer le fond lié à la réflexion parasite en d'en calculer une moyenne temporel. Ce fond est soustrait à l'image courante (soustraction point à point de l'image brute avec la moyenne temporelle du fond). Cela permet d'améliorer le contraste de l'image courante. L'utilisateur doit être formé pour acquérir le fond. A l'heure actuelle la soustraction n'est valide que lorsque les paramètres d'acquisition du fond et de l'objet sont identiques (problème d'offset électronique, méconnaissance de la courbe de gain). **On reparle du talon ? : influence par le module de visualisation ?**.
- **Correction du décalage inter-ligne** : Il s'agit d'un mode utilisé par défaut (groupe paramètres électroniques). **Evolution avec la nouvelle électronique et le nouveau boîtier ?**
- **Moyennage temporel** : Aujourd'hui seul ce type de moyennage peut être effectué, est-il judicieux de laisser la possibilité à l'utilisateur d'appliquer d'autres types de moyennages ?
- **Correction automatique du gain** : Il est encore peut utiliser aujourd'hui du fait des variations de signal lié à l'offset électronique. Il peut faire perdre une image ou deux lors de l'acquisition. **Evolution avec le nouveau boîtier ?**
- **Détection des fibres, calibration, reconstruction...** : scénario, interface et implémentation ? **Choix avec ou sans détection des fibres ?**

### 6.2 TREAI

- **Détection des fibres, calibration, reconstruction...** : scénario, interface et implémentation ?
- **Correction des distorsions géométriques** : On peut envisager un module de traitement dans lequel on rentre les informations nécessaires (cf. programme prototype) et une image de la mire de distorsion. Un champ de déformation est alors calculé (en X, en Y). Ce champ calculé ainsi que les informations nécessaires sont alors utilisées pour supprimer les distorsions sur l'image d'intérêt. On peut alors appliquer les outils spécifiques de mesures (interactifs) à l'image corrigée.

## 7 Question ouvertes

Module ActiveX ou JavaBeans pour l'intégration de nos modules dans des applications tiers ?
Export/Accès par le net ?
Valeur ajoutée ?

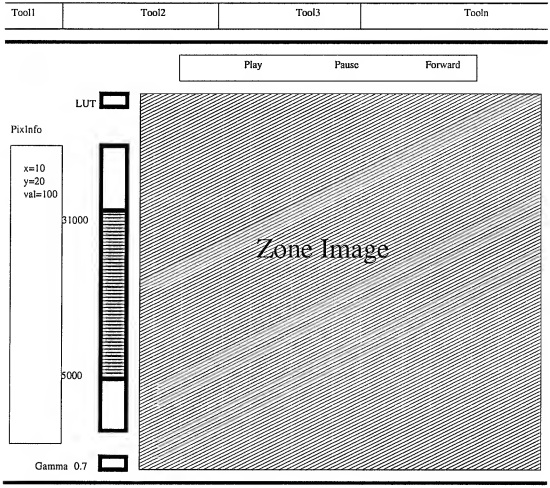
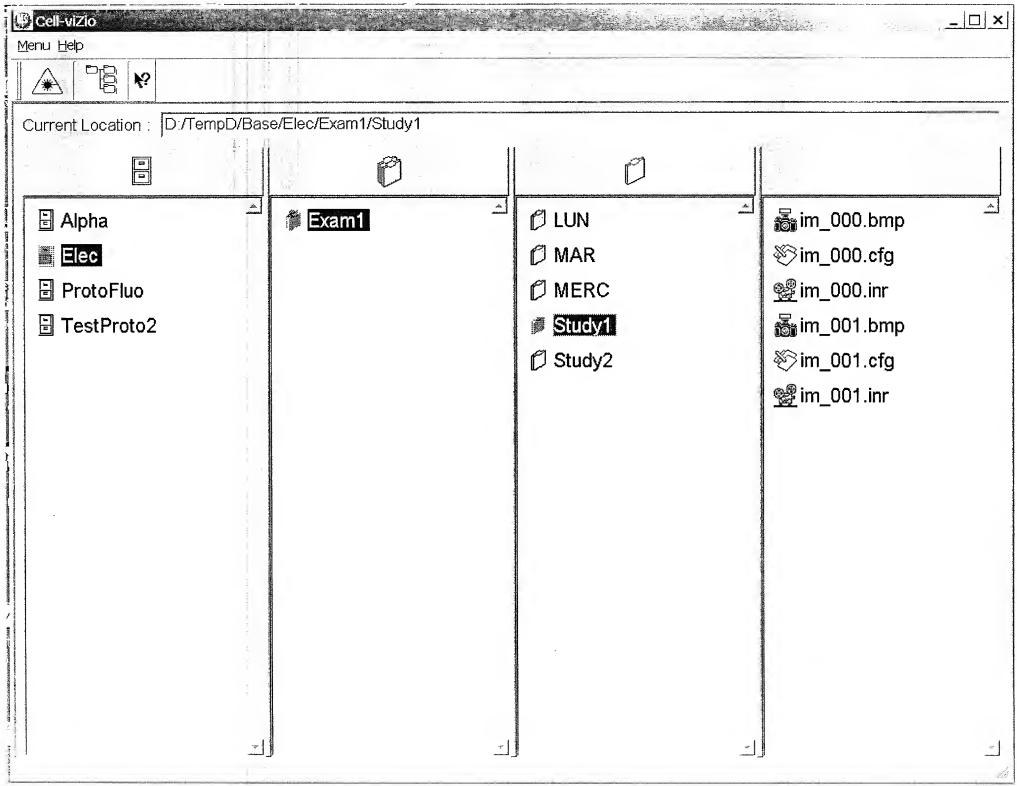
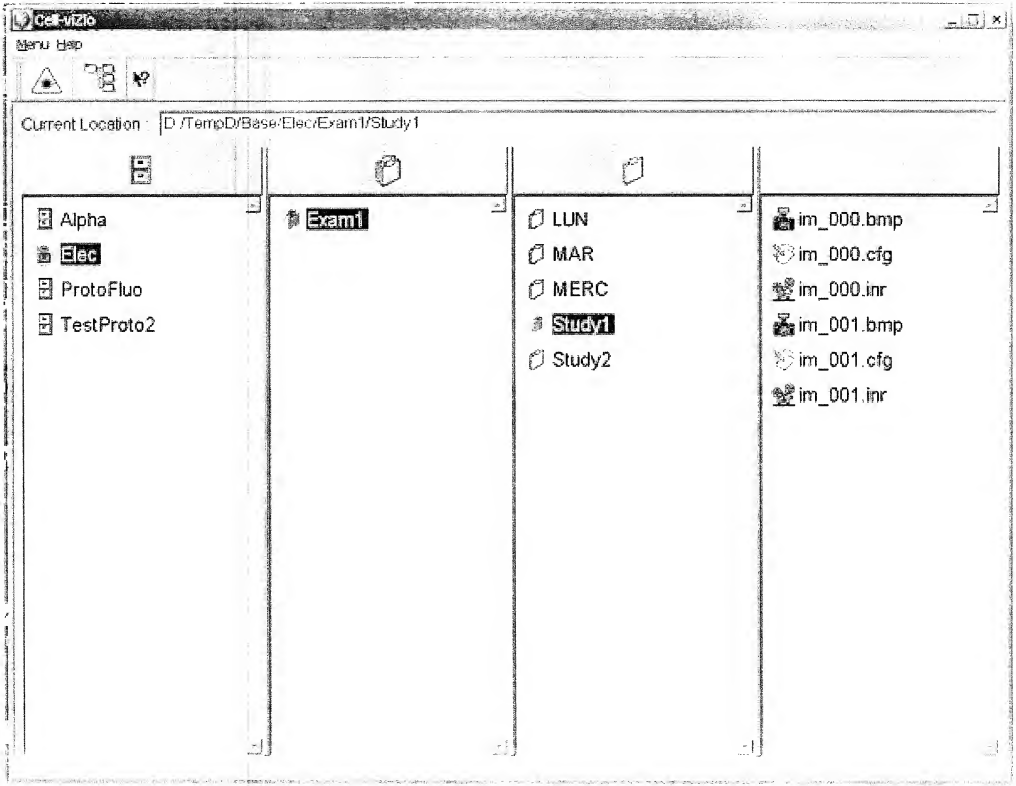



FIG. 1 – modVisu

**Diligence Document 4**


Start / Stop
Pause/Continue


Cell-viZio
Menu Help
Current Location : D:/TempD/Base/Elec/Exam1/Study1
Alpha
Elec
ProtoFluo
TestProto2
Exam1
LUN
MAR
MERC
Study1
Study2
im_000.bmp
im_000.cfg
im_000.inr
im_001.bmp
im_001.cfg
im_001.inr


Cell-Vizio
Menu Help
Current Location
D:/TempD/Base/Elec/Exam1/Study1
Alpha
Elec
ProtoFluo
TestProto2
Exam1
LUN
MAR
MERC
Study1
Study2
im_000.bmp
im_000.cfg
im_000.inr
im_001.bmp
im_001.cfg
im_001.inr

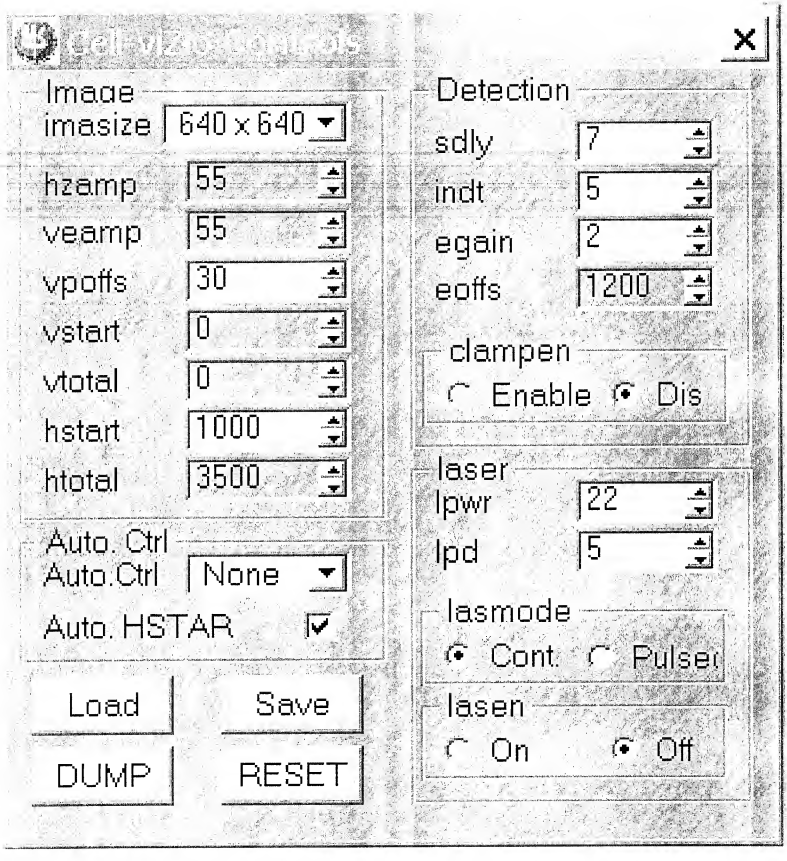


Image
imasize
640 x 640
hzamp
55
veamp
55
vpoffs
30
vstart
0
vtotal
0
hstart
1000
htotal
3500
Auto. Ctrl
Auto.Ctrl
None
Auto. HSTAR
Load
Save
DUMP
RESET
Detection
sdly
7
indt
5
egain
2
eoffs
1200
clampen
Enable
Dis
laser
lpwr
22
lpd
5
lasmode
Cont.
Pulse
lasen
On
Off

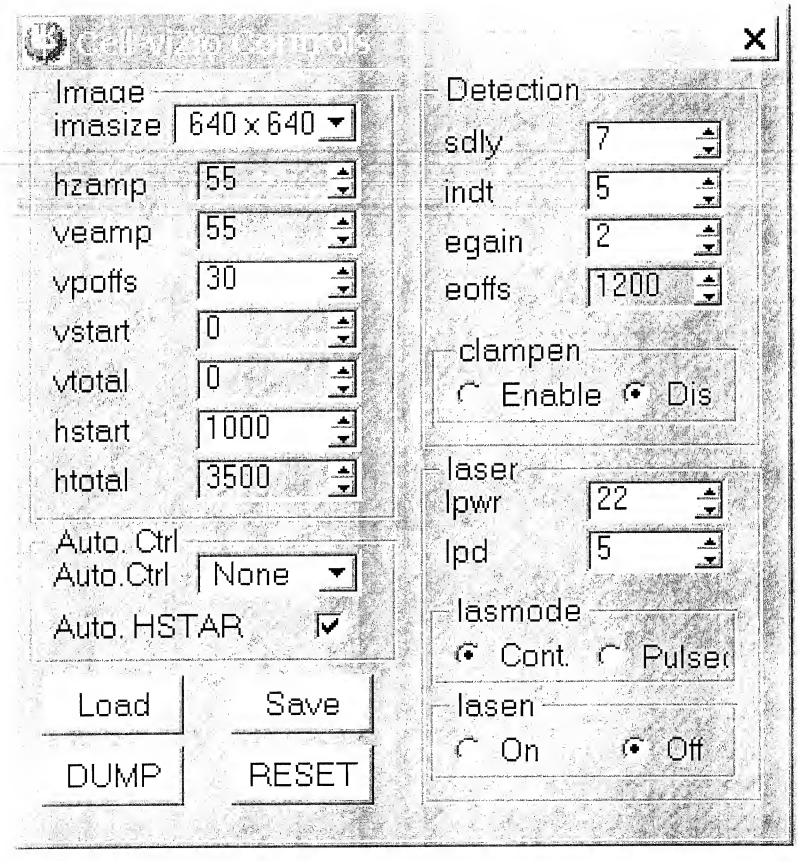


Image
imasize
640 x 640
hzamp
55
veamp
55
vpoffs
30
vstart
0
vtotal
0
hstart
1000
htotal
3500
Auto. Ctrl
Auto.Ctrl
None
Auto. HSTAR
Load
Save
DUMP
RESET
Detection
sdly
7
indt
5
egain
2
eoffs
1200
clampen
Enable
Dis
laser
lpwr
22
lpd
5
lasmode
Cont.
Pulse
lasen
On
Off

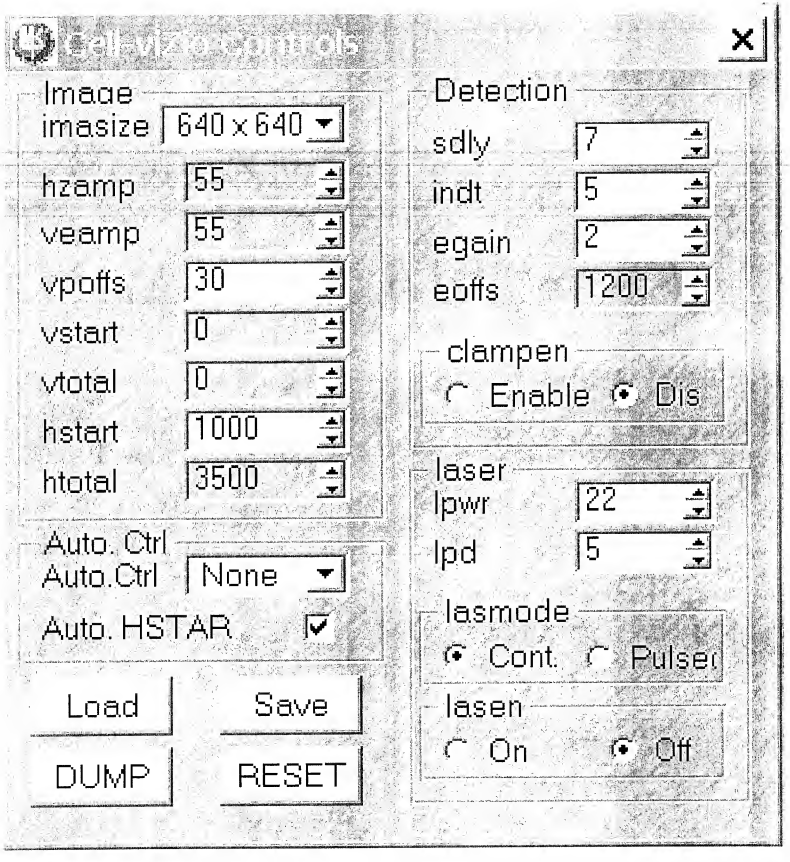
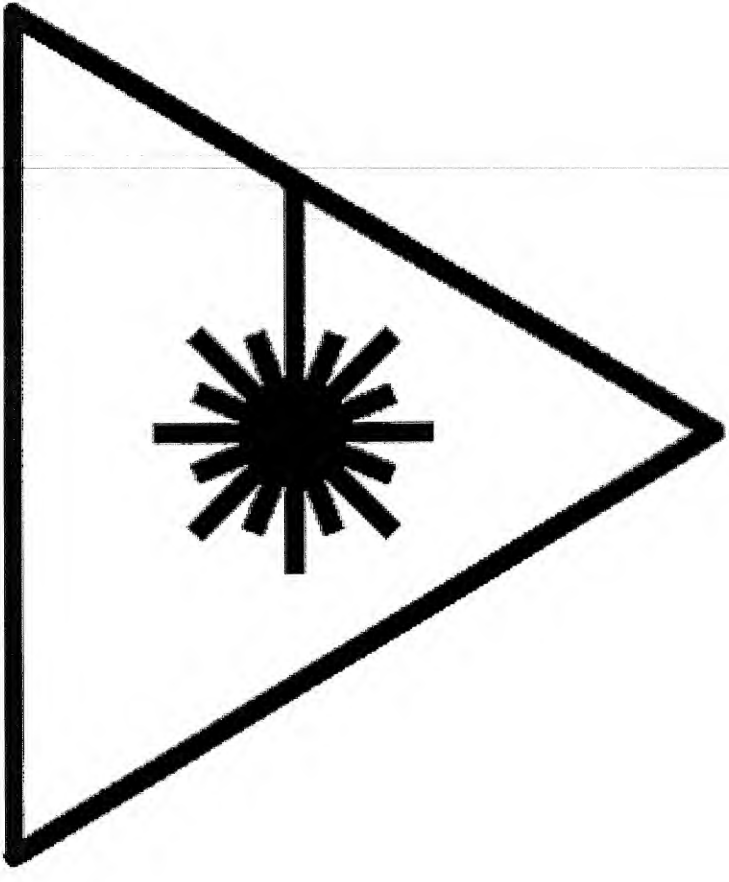
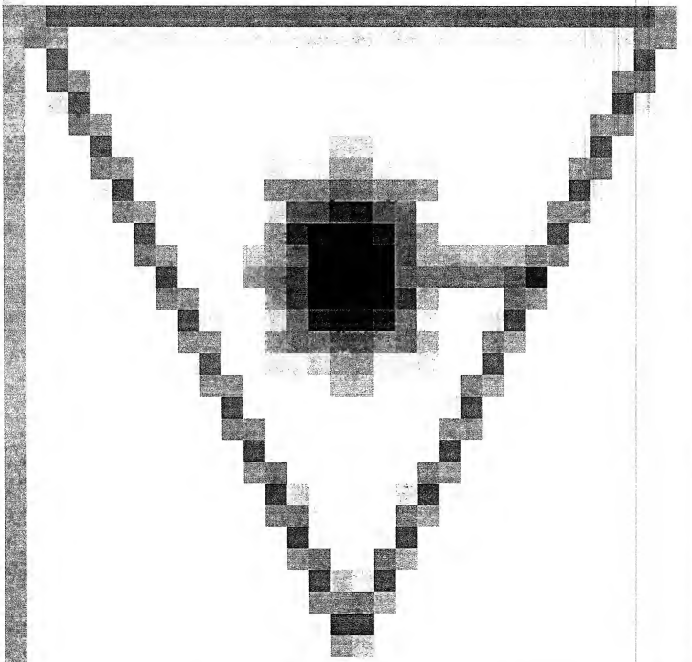
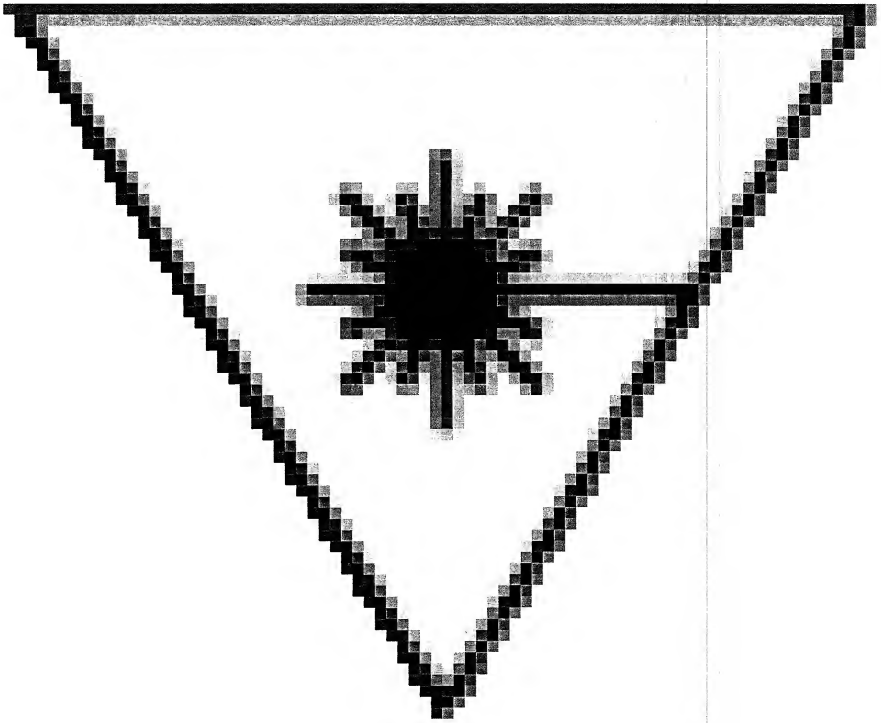
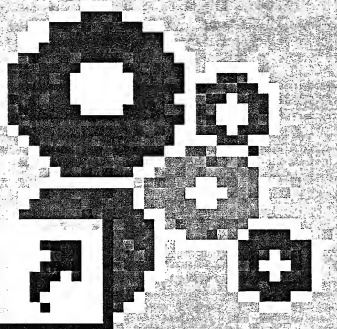


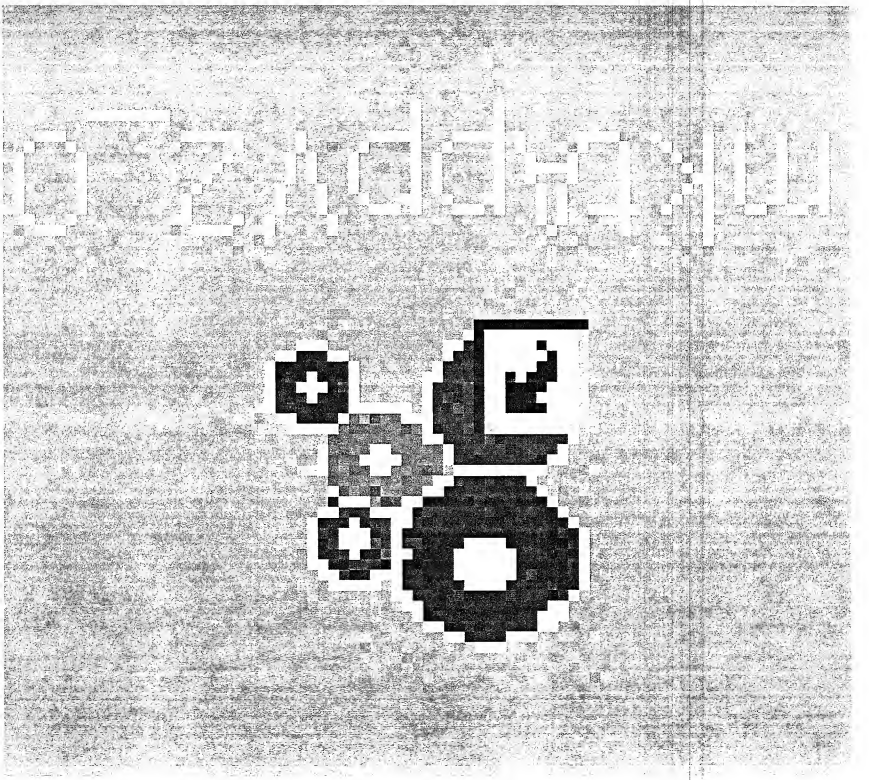
Image
imasize
640 x 640
hzamp
55
veamp
55
vpoffs
30
vstart
0
vtotal
0
hstart
1000
htotal
3500
Auto. Ctrl
Auto.Ctrl
None
Auto. HSTAR
Load
Save
DUMP
RESET
Detection
sdly
7
indt
5
egain
2
eoffs
1200
clampen
Enable
Dis
laser
lpwr
22
lpd
5
lasmode
Cont.
Pulse
lasen
On
Off

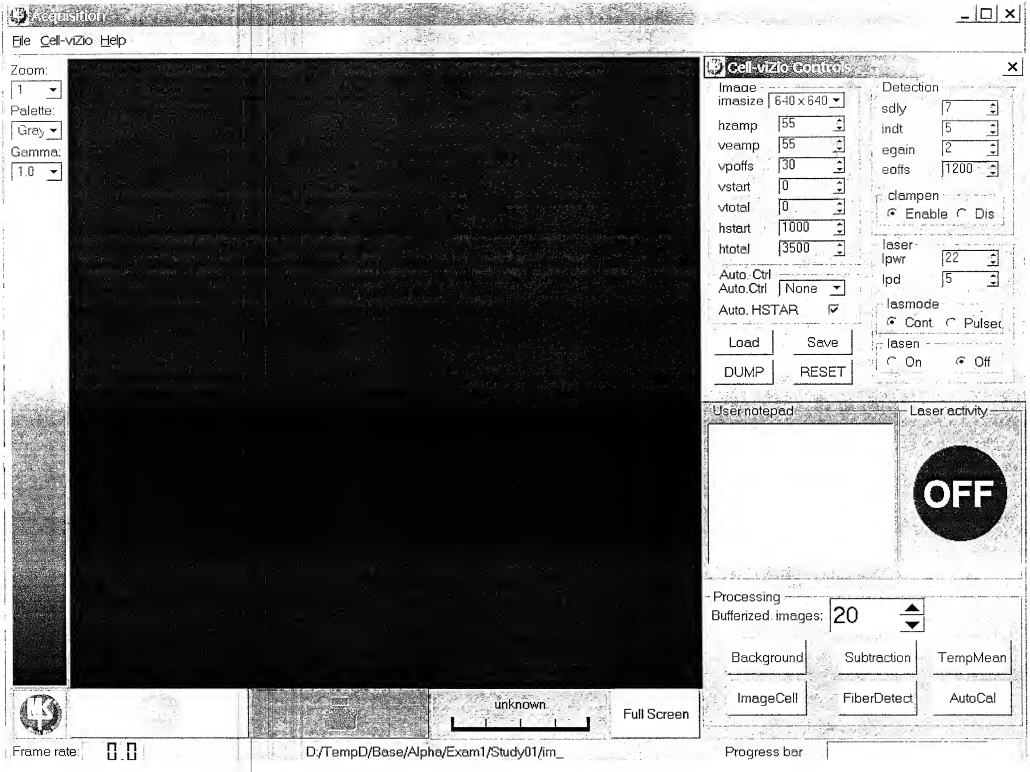






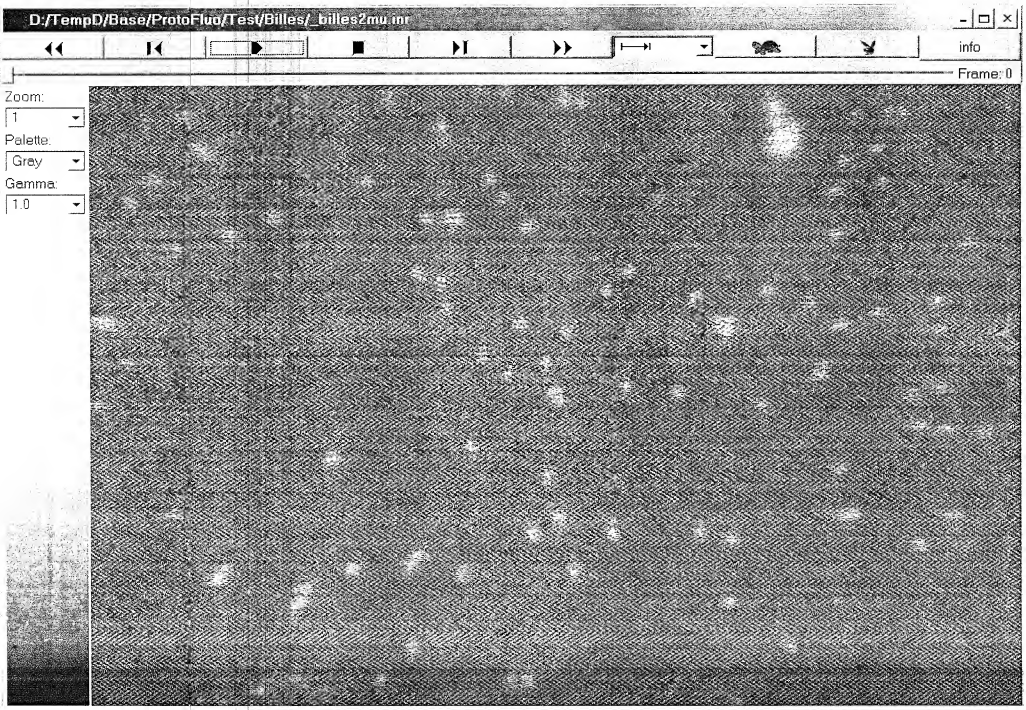


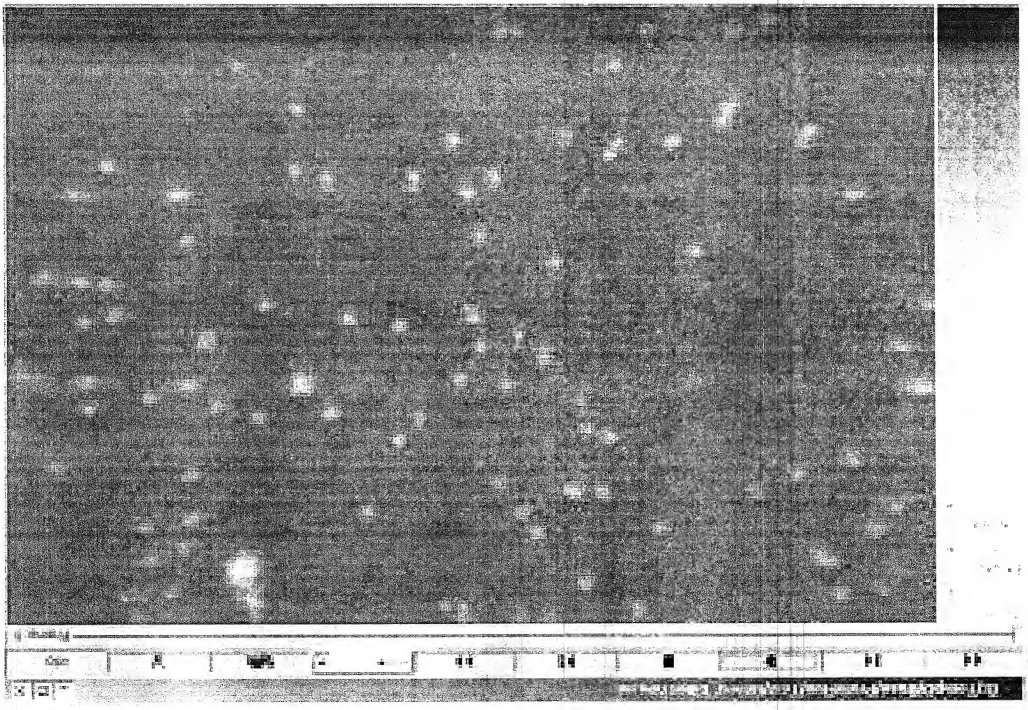


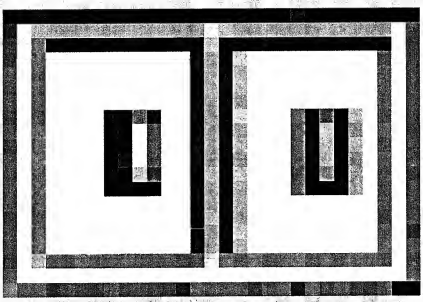

Acquisition
File Cell-viZio Help
Zoom:
1
Palette:
Gray
Gamma:
1.0
Cell-viZio Control
Image
imasize 640 x 640
hzamp 55
veamp 55
vpoffs 30
vstart 0
vtotal 0
hstart 1000
htotal 3500
Auto Ctrl
Auto.Ctrl None
Auto. HSTAR
Load
Save
DUMP
RESET
Detection
sdly 7
indt 5
egain 2
eoffs 1200
clampen
Enable
Dis
laser
lpwr 22
lpd 5
lasmode
Cont.
Pulsed
lasen
On
Off
User notepad
Laser activity
OFF
Processing
Bufferized images: 20
Background
Subtraction
TempMean
ImageCell
FiberDetect
AutoCal
unknown
Full Screen
Frame rate: 0.0
D:/TempD/Base/Alpha/Exam1/Study01/im_
Progress bar

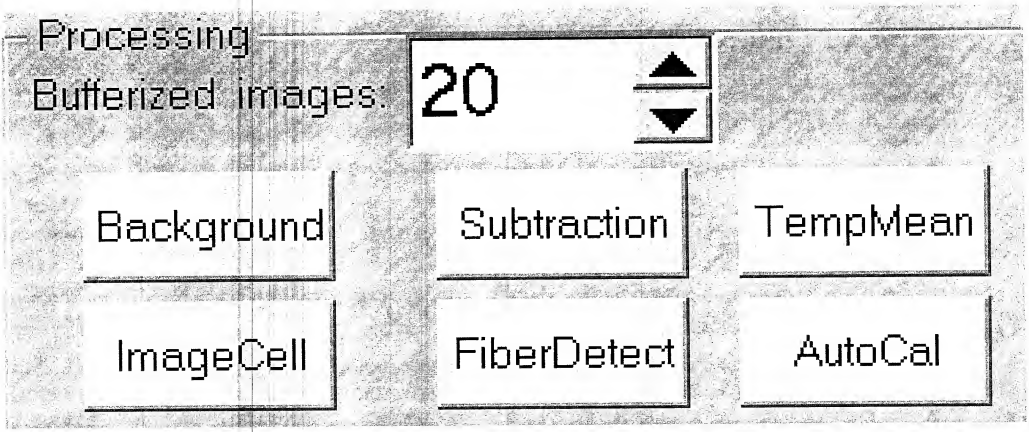
[redacted]

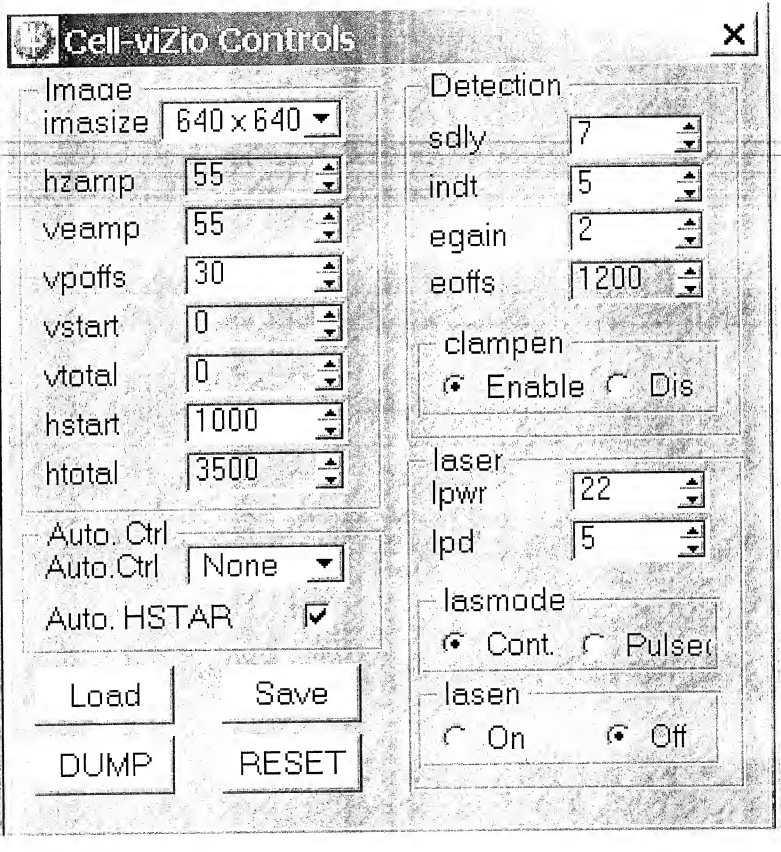
OFF

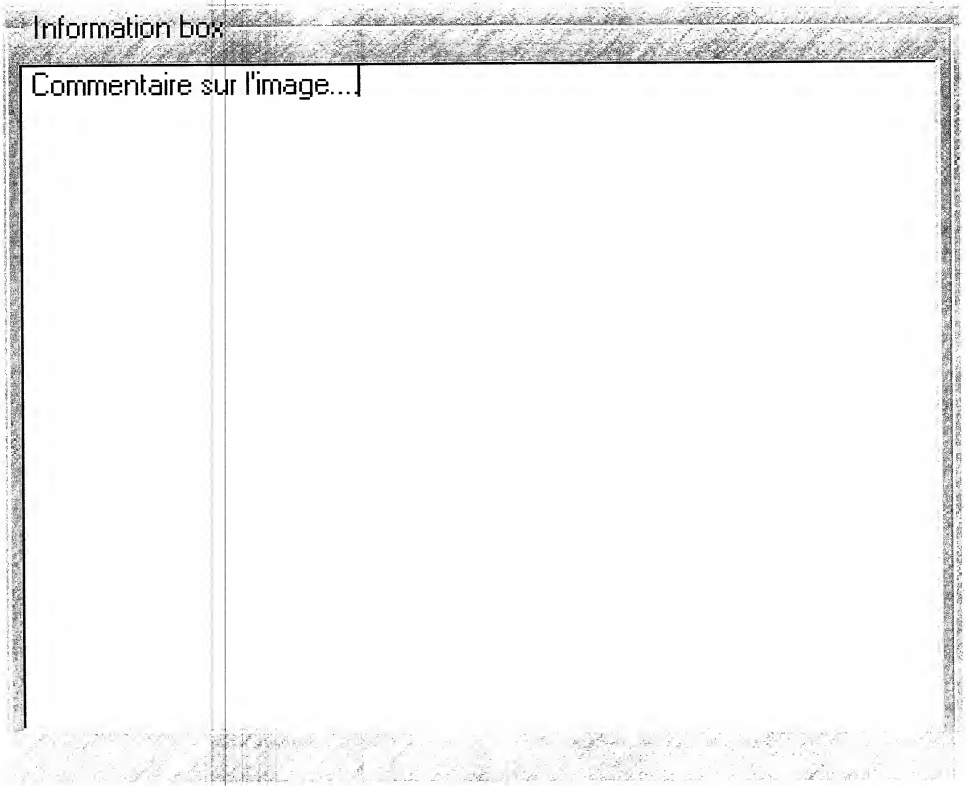

D:/TempD/Base/ProtoFluo/Test/Billes/_billes2mu.inr
info
Frame: 0
Zoom:
1
Palette:
Grey
Gamma:
1.0

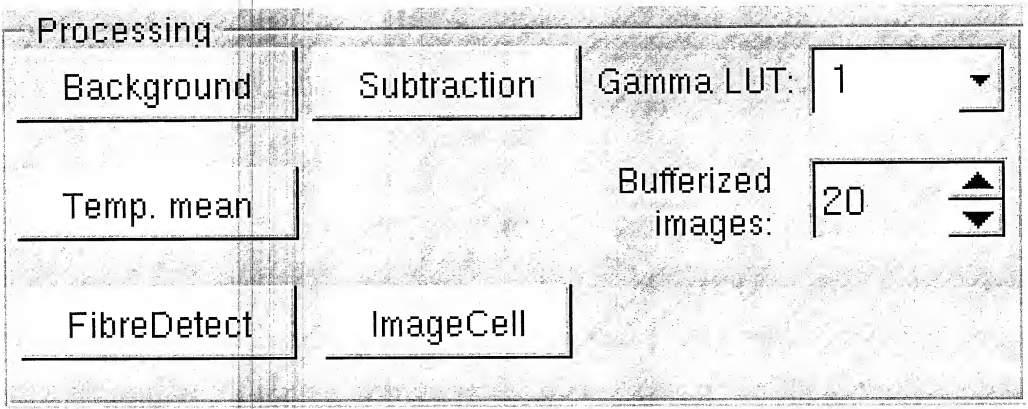


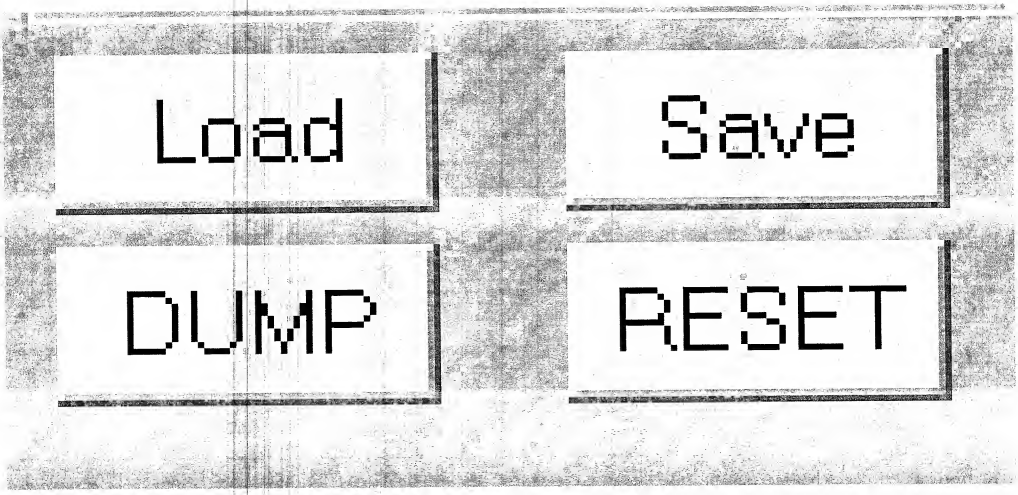



Processing
Bufferized images:
20
Background
Subtraction
TempMean
ImageCell
FiberDetect
AutoCal

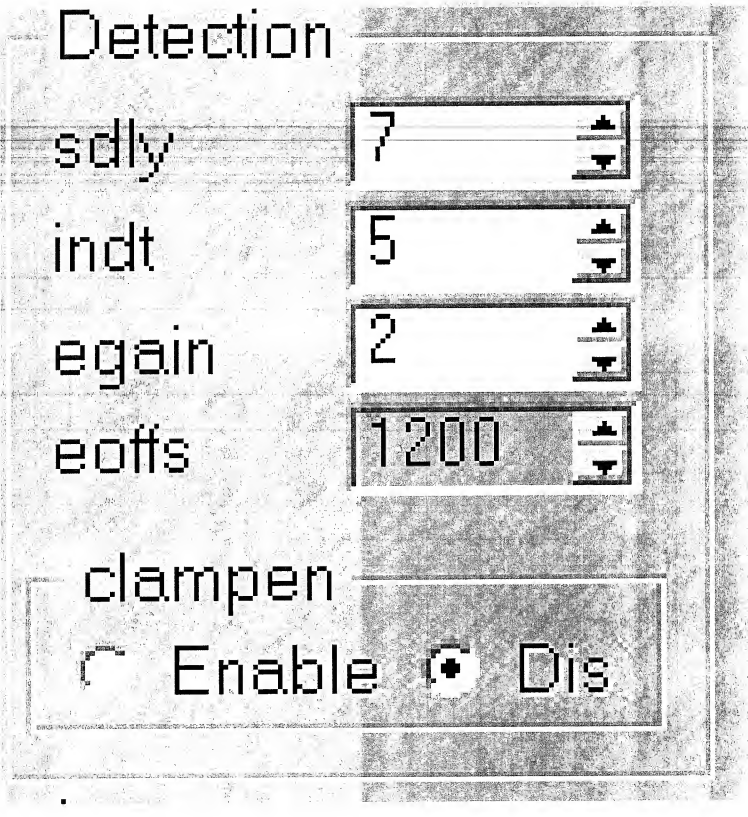

Cell-viZio Controls
Imade
imasize
640 x 640
hzamp
55
veamp
55
vpoffs
30
vstart
0
vtotal
0
hstart
1000
htotal
3500
Auto. Ctrl
Auto.Ctrl
None
Auto. HSTAR
Load
Save
DUMP
RESET
Detection
sdly
7
indt
5
egain
2
eoffs
1200
clampen
Enable
Dis
laser
lpwr
22
lpd
5
lasmode
Cont.
Pulse
lasen
On
Off


Information box
Commentaire sur l'image...


Processing
Background
Subtraction
Gamma LUT:
1
Temp. mean
Bufferized images:
20
FibreDetect
ImageCell


Load
Save
DUMP
RESET

Send
Load
Save


Detection
sdly
7
indt
5
egain
2
eoffs
1200
clampen
Enable
Dis

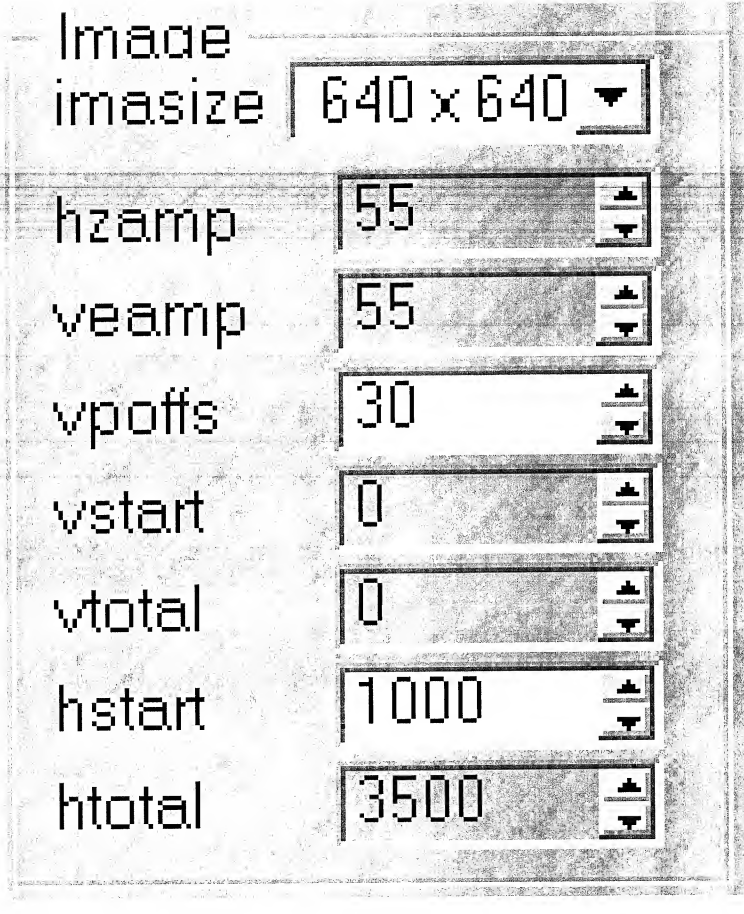
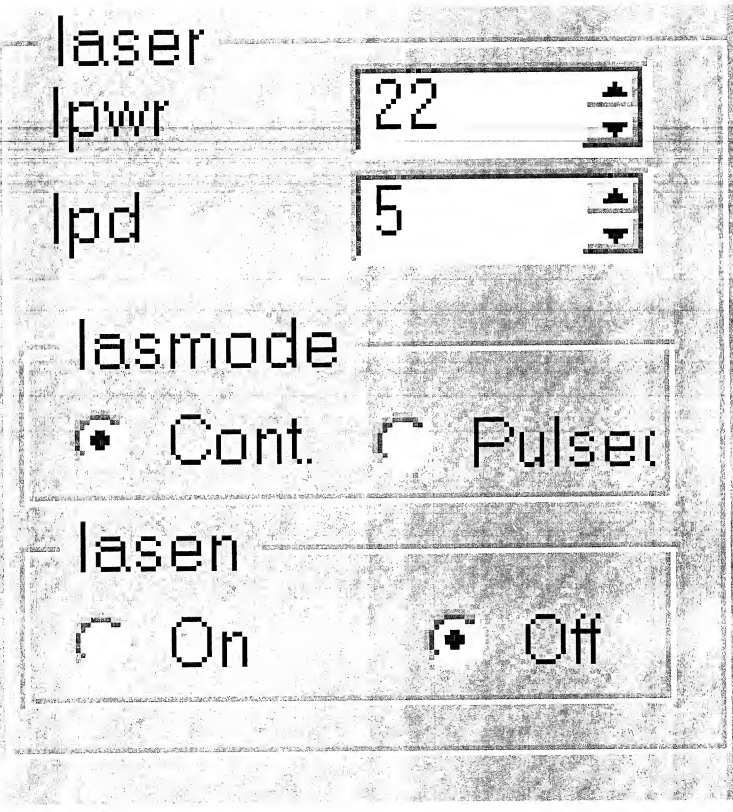
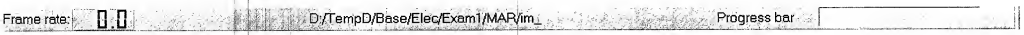
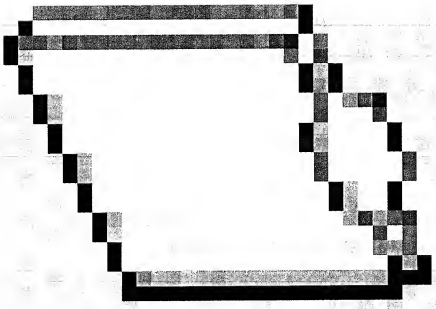


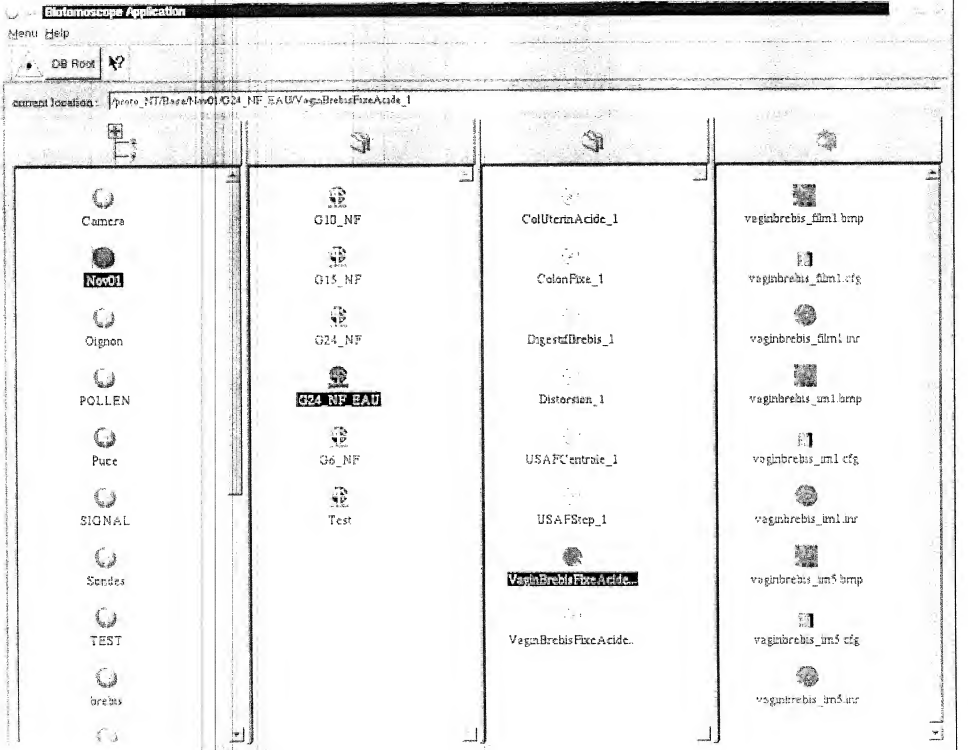
Image
imasize
640 x 640
hzamp
55
veamp
55
vpoffs
30
vstart
0
vtotal
0
hstart
1000
htotal
3500


laser
lpwr
22
lpd
5
lasmode
Cont.
Pulse
lasen
On
Off


unknown
Full Screen


Frame rate: 0.0
D:/TempD/Base/Elec/Exam1/MAR/im_
Progress bar




Biofomoscope Application
Menu Help
DB Root
current location : /proto_NT/Base/Nov01/G24_NF_EAU/VaginBrebisFixeAcide_1
Camera
Nov01
Oignon
POLLEN
Puce
SIGNAL
Sondes
TEST
brebis
G10_NF
G15_NF
G24_NF
G24_NF_EAU
G6_NF
Test
ColUterinAcide_1
ColonFixe_1
DigestifBrebis_1
Distorsion_1
USAFCentrale_1
USAFStep_1
VaginBrebisFixeAcide...
VaginBrebisFixeAcide..
vaginbrebis_film1.bmp
vaginbrebis_film1.cfg
vaginbrebis_film1.inr
vaginbrebis_im1.bmp
vaginbrebis_im1.cfg
vaginbrebis_im1.inr
vaginbrebis_im5.bmp
vaginbrebis_im5.cfg
vaginbrebis_im5.inr

**Diligence Document 5**

# Manuel de référence du module ImageCell

Auteur : Aymeric Perchant

Date : 2002-02-07

**Diffusion :** interne

**Section :** informatique, image

**Sujet :** Manuel de référence de la partie libMKTProcessing pour imageCell. Ce manuel contient les informations de traitement d'images, de programmation C++, et d'utilisation.

**Version :** *Revision* : 1.1

**Référence du document :** `$Id: ReferenceImageCell.tex,v 1.1 2002-02-07 16:30:08 aymeric Exp $`

## Introduction

Ce document rassemble les différentes informations de référence pour le module principal intitulé Image-Cell. Ce module contient les parties suivantes de traitement d'images :

- détection des fibres,
- estimation des flux revenant des fibres,
- corrections des défauts de l'appareil (biais, fond, ...),
- calibration de l'appareil,
- reconstruction d'images.

Ces différentes parties seront abordées en détail dans chaque partie pour expliquer les algorithmes et les paramètres (chapitre 1), l'implantation de ceux-ci (2) et leur utilisation pratique (3).

# Table des matières

# Chapitre 1

# Traitement d'images dans Image-Cell

## Introduction

Cette partie décrit les algorithmes en commençant par la description générale, jusqu'aux boites élémentaires.

## 1.1 Schéma global

La figure 1.1 représente le schéma global du traitement. On peut distinguer quatre blocs, dont deux sont identiques, à une paramétrisation près. Il existe donc trois groupes de traitements qui sont les suivants :

**Détection des fibres :** ces traitements permettent de détecter et d'isoler chaque fibre sur une image, ainsi que d'analyse la structure d'agencement des fibres du guide d'images;

**Estimation des flux :** une image brute de taille $640 \times 640 \times 20$ (largeur, hauteur, nombre d'images temporellement )contient 8 méga pixels qui représentent l'information vue par 10000 ou 30000 fibres. Ce bloc permet d'isoler l'information effectivement vue par chaque fibre;

**Calibration et reconstruction :** La connaissance de l'information vue par chaque fibre est ensuite traitée pour être reconstruite sous la forme d'une image débarrassée des défauts de l'appareil.

Chacun de ces trois blocs sont maintenant détaillés. Le prototypage des algorithmes décrits ici est détaillé dans le rapport de stage de Sandra Marti [?]; nous renvoyons le lecteur à cette référence pour plus de renseignements sur la démarche de développement des algorithmes, notamment celui de détection des fibres.
Le bloc de gauche d'estimation des flux permet de mesurer le flux sur une image qui représente un objet aux propriétés constantes dans l'espace (un milieu diffusant homogène ou un objet presque homogène en mouvement aléatoire, un miroir). La sortie de ce bloc est donc une image des taux d'injection dans chaque fibre, et qui en prend en compte l'intégralité de la chaîne.
Le bloc de droite d'estimation des flux permet de l'estimer pour un objet à observer, et à travers l'appareil. Le dernier bloc de calibration est une division de l'image des taux d'injection par l'image de l'objet observé. Cette opération permet de compenser les mauvaises injections dans certaines fibres. Puis l'image est reconstruite.

## 1.2 Bloc de détection des fibres

La détection des fibres s'organise autour de quatre traitements successifs :
- prétraitements,

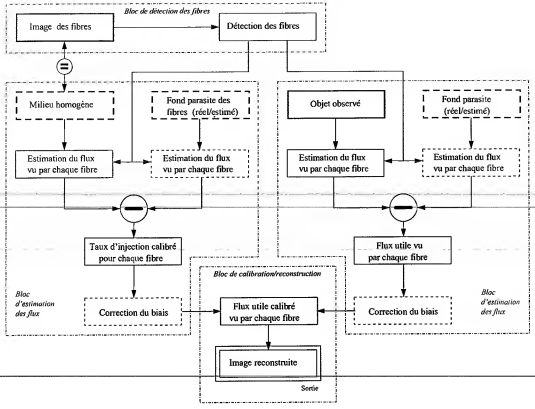



FIG. 1.1 – Schéma global du traitement. Les cadres gras représentent les entrées. Les cadres pointillés représentent les données ou les traitements facultatifs (biais, soustraction du fond...).

- ligne de partage des eaux (LPE) : la première segmentation par région,
- séparation des fibres trop grosses (split),
- fusion des fibres trop petites (merge).

Les deux dernières étapes peuvent être bouclées ; néanmoins une analyse de performance et de convergence doit alors être menée. Nous n'aborderons pas ce sujet pour le moment.

### 1.2.1 Prétraitements

Les prétraitements réalisés sont les suivants.

**Diffusion anisotrope :** le but est de lisser l'image dans les zones plates, c'est à dire les zones inter fibres. Le prototype fonctionnait mais n'utilisait pas la librairie inrimage, mais une autre librairie (celle utilisée par Sandra, issue de l'ENST). Le réglage de la diffusion est laissée à plus tard, lors de la prochaine release de la libInrimage qui contiendra de la documentation à ce sujet ! Paramètre `[_doDiffusion_]`.

**Interpolation $\times 2$ au plus proche voisin :** on cherche à simuler des éléments structurant de morphologie mathématique avec un rayon inférieur à un. L'image est doublée pour que l'ouverture qui suit ne touche pas aux maxima isolés, mais seulement ceux qui sont 8-connexe, mais non 4 connexe (voisins par une diagonale). L'intérêt est de faire une sélection des maxima éliminé par l'ouverture. Paramètre `[_doZoom_]`.

**Ouverture numérique :** on cherche à éliminer les maxima parasites situés sur les fibres. C'est un prétraitement classique de la LPE, mais éventuellement modifié par le comportement du doublage de l'image expliqué précédemment.

**Inversion de l'image :** la LPE fonctionne à partir de minima, on inverse donc l'image pour transformer les minima en maxima.

L'opération d'interpolation et d'ouverture qui suit devrait pouvoir être remplacé par un nouvel opérateur de morphologie mathématique qu'il faudrait définir. Cela permettrait un gain important, puisque toutes les opérations suivantes se feraient sur une image de taille inférieure.

### 1.2.2 Premier watershed

Le watershed de la libInrimage nécessite la définition de marqueurs dans l'images. Nous obtenons les marqueurs comme minima de l'image prétraitée avec de h-dômes [?]. Les h-domes pour les minima sont définis pour un entier $h$, une fonction $f : \mathbb{Z}\!\!\mathbb{Z} \times \mathbb{Z}\!\!\mathbb{Z} \rightarrow \mathbb{Z}\!\!\mathbb{Z}$ représentant l'image concernée :

$$g = E_f^{\infty}(f + h) - f, \tag{1.1}$$

avec $E_a^{\infty}(b)$ la reconstruction géodésique par érosion de $b$ dans $a$. $g$ désigne ici l'ensemble des h-domes, que l'on seuil à 1 afin de tous les récupérer. On obtient ainsi les minima de profondeur au moins égale à $h$. Ce paramètre $h$ est donné par `[_hDomeWatershed_]`.
Le résultat de cette LPE est une image de composantes connexes représentant chaque fibre détectée. Afin de résoudre bon nombre de problèmes de bord on colle les composantes touchant le bord au bord lui-même. Ces composantes sont considérées comme faisant partie du bord, et du fond et sont mise à la composante connexe d'indice 0, qui est l'indice réservé au fond, et au bord.
Enfin, les composantes connexes sont réorganisées pour ne pas présenter de trous, et sont triés par ordre décroissant de taille.

### 1.2.3 Split

Les prétraitements, et notamment l'ouverture numérique servent à limiter le défaut principal de la LPE qui est de sursegmenter. En faisant cela, on a tendance à augmenter le nombre de segments qui vont englober plusieurs objets (ici, des fibres). Le premier défaut que l'on corrige est la sous-segmentation inévitable au traitement, mais surtout rajoutée par ces prétraitements. Pour cela on effectue les traitements suivants.

**Sélection des segments suspectés de sous-segmentation :** on trouve les fibres dont la taille est supérieure à un seuil fixé en fonction de la moyenne des tailles normalisées (paramètre `[_oversizedFactor_]`), et en fonction du nombre de voisins (paramètre `[_overSizedNeighLimit_]`).

**Re-segmentation de ceux-ci :** Les segments sélectionnés sont isolées et re-segmenter soit sur l'image initiale sans prétraitement, soit sur l'image de carte de distance à l'intérieur de ces régions (paramètre `[_overSizedResegmentMethod_]`). Dans le premier cas, on suppose que les défauts majoritaires proviennent des prétraitement, dans le second cas on suppose qu'ils étaient là avant, et que l'on souhaite séparer les segment aux endroits de rétrécissement (forme de la cacahuète). La LPE est effectuée avec les même paramètres que la première.

### 1.2.4 Merge

Puis on cherche à corriger le défaut le plus classique (pour une LPE) : la sursegmentation. On cherche donc à fusionner les segments. Pour cela on effectue les opérations suivantes :
- présélection des *candidats à la fusion*,
- parmi ceux-ci trier ceux qui seront obligatoirement fusionnés,
- pour les autres, éliminer les fusions impossibles
  - répertorier les fusions possibles

- éliminer les fusions donnant de mauvais résultats
  si il ne reste pas de fusion après filtrage, retirer la fibres des fibres à fusionner
- pour toutes les fibres restantes, prendre la meilleur fusion au sens de la compacité des segments fusionnés.

Les points importants sont détaillés ci-après.

### Remarques sur la compacité

La compacité utilisée est le critère le plus simple existant. Pour un objet de périmètre $P$ et de surface $S$, le critère est $\frac{P^2}{S}$. Le périmètre est pris comme étant la frontière de l'objet 4-connexe, et donc est défini comme tous les pixels de l'objet ayant un voisin 8-connexe avec le fond (défini ici comme étant tout le reste de l'image. La surface est simplement l'objet lui-même.
Cependant, ce critère simple à calculer n'est pas invariant avec l'échelle de l'objet. On peut montrer que pour des formes simple, le critère est croissant avec la taille de l'objet au lieu d'être invariant. Cette croissante n'est cependant qu'en moyenne en $\frac{1}{n^2}$, avec n la diamètre de l'objet (ici, pour une boule en 4-connexité). Ce critère reste donc tout de même assez raisonnable, mais plus délicat à manier dans le cas de la comparaison de la compacité d'objets de taille différentes.
Il faut remarquer ici que les premiers essais sont réalisés sur une image déformée par les distortions, et qu'il faudrait, pour être plus précis soit corriger l'image avant, soit tenir compte de l'anisotropie non stationnaire spatialement : on a tendance à surestimer à la fois les frontières et la surface sur les bords de l'image.

### Sélection des fibres détectées trop petites

La sélection s'effectue en deux étapes. On commence par trouver les segments candidats à la fusion avec un seuil sur la taille. On utilise le paramètre `[_underSizedFactor_]` qui est un facteur multiplicatif de la moyenne, et est donc inférieur à 1 en général. La seconde étape est de trouver parmi ces segments lesquels doivent obligatoirement être fusionnés, et lesquels doivent être filtrés.

### Les segments obligatoirement fusionnés

La sélection des segments (ou fibres) obligatoirement fusionnés s'effectue dans l'espace (nombre de voisins) $\times$ (taille normalisée). La taille est normalisée par rapport à la moyenne (on divise toutes les tailles par la moyenne des tailles). Le nombre de voisins est compté avec la 8-connexité.
Les figures 1.2 et 1.3 représentent l'histogramme conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. On remarque qu'il existe un axe principal qui est centré sur la droite passant par des points d'intérêts particuliers $(6, 1)$, $(8, 2)$, et $(5, .5)$. Ces points peuvent s'explique géométriquement, comme indiqué sur la figure 1.4
Cet axe est donc un axe naturel de la structure hexagonale des fibres dans cet espace. En s'orientant sur cet axe, les petites fibres se situent vers un nombre de voisins faibles et une taille faible. On détermine une frontière linéaire pour sélectionner les fibres détectées réellement trop petites. Après quelques tests sur des images réelles de la base de données «Novembre 2001» et en fonction des résultats préliminaires de [?], on a choisi une droite passant par les points $(6; \frac{1}{3})$ $(3; 1)$. Les paramètres de cette droites sont données sous la forme $y = mx + p$, avec $m$ la pente `[_underSizedMendatoryMergeSlope_]` et $p$ l'ordonnée à l'origine `[_underSizedMendatoryMergeOffset_]`.
Tout point en dessous de la droite sera fusionné obligatoirement, et ne va donc pas passer par les filtres décrits dans la section suivante. Les autres fibres sont par contre filtrée comme suit.

### Filtrage des fusions possibles

On entend par filtrage ici un procédé qui retire certaines fusions. Il existe trois filtres possibles.

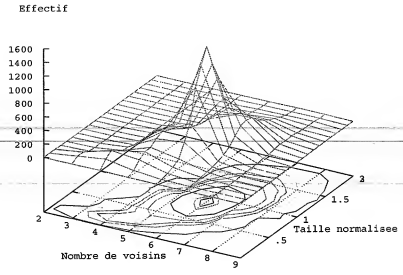



FIG. 1.2 – Vue 3d de l'histogramme 2d conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. Des lignes de niveaux sont projetées sur la fond du graphique : les lignes sont centrées autour de 6 voisins et de la moyenne de la taille des fibres.

**Filtrage sur la taille de la fusion :** (paramètre `[_underSizedFilterSize_]`) on élimine les fusions dont la taille totale dépasse un seuil fixé par le paramètre `[_underSizedFilterSizeMax_]`, qui est pris comme facteur multiplicatif de l'écart type ajouté à la taille moyenne.

**Filtrage sur la compacité totale :** (paramètre `[_underSizedFilterCompact_]`) on élimine les fusions dont la compacité totale dépasse un seuil fixé par le paramètre `[_underSizedFilterCompactMax_]`, qui est pris comme facteur multiplicatif de l'écart type ajouté à la taille moyenne.

**Filtrage sur le changement de compacité :** (paramètre `[_underSizedFilterMoreCompact_]`) on élimine les fusions qui augmentent la compacité de l'objet avec lequel on veut fusionner.

Dans les paramètres par défaut, on n'utilise pour le moment que le premier filtre qui semble suffisant dans la plupart des cas. Les autres filtres semblent trop restrictifs et doivent être encore étudiés.

## 1.3 Bloc d'estimation des flux

Le bloc d'estimation des flux comprend plusieurs sous-parties :

**estimation du flux vu par chaque fibre :** c'est ce que l'on peu récupérer d'une image à l'aide de la détections es fibres ;

**estimation du fond de l'image :** ici le fond désigne les réflexions parasites ou (*ou* inclusif !) l'offset du à l'électronique et au détecteur ;

**soustraction du fond :** on retire le fond à l'image ;

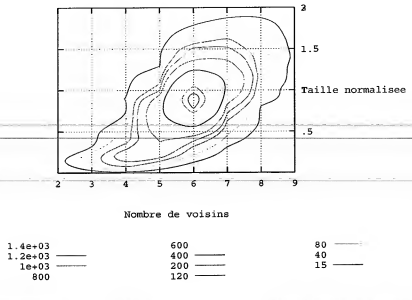



FIG. 1.3 – Lignes de niveau de l'histogramme 2d conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. La forme 3d de cet histogramme est donnée sur la figure 1.2

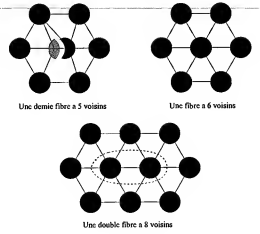



FIG. 1.4 – Variation du nombre de voisins en fonction de la taille du segment détecté comme étant une fibre

**correction du biais :** les images sont en général biaisés, c'est à dire qu'il existe un fond lentement variable au lieu d'être constant.

Le bloc de soustraction du fond est optionnel, ainsi que celui de la correction du biais.

### 1.3.1 Estimation du flux vu par chaque fibre

Le rapport [?] décrit une méthode pour ***rattraper*** les problèmes de saturation de l'image. Ces problème vont disparaître avec l'arrivée de l'électronique mi mai 2002. Nous avons donc décidé de ne pas continuer dans cette voie pour le moment. Ici nous avons simplement choisi d'estimer le flux avec la moyenne sur la fibre.
Plusieurs améliorations sont envisageables :
- tenir compte du bruit poissonien dans l'estimateur du max avec la moyenne,
- tenir compte de la disparité de forme de fibres du guide d'image ; on peut constater que les toutes les fibres ne sont pas identiques et que l'estimateur du flux devrait s'adapter à la forme exacte de la fibre,
- vérifier les problèmes possibles dues à la zone inter-fibre, en fonction de la tache focale sur l'entrée du guide,
- et plus généralement récupérer le modèle complet d'injection et de retour du spot décrit dans [?].

Ces améliorations peuvent être étudiées, mais ne sont pas prioritaires pour le moment.

### 1.3.2 Estimation ou utilisation du fond, puis soustraction

Le fond peut avoir plusieurs sources ; celles-ci sont décrite dans les mémos [?, ?, ?]. En résumé, nous appelons fond ici soit les réflexions parasites sur les optiques, et donc y compris sur la sortie du guide d'images, mais également l'offset due à la chaîne de numérisation.
Si l'offset est dominant sur l'image, on ne peut pas obtenir le fond simplement en retirant l'image, car l'offset dépend du contenu, et n'est donc plus le même. Dans ce cas, on utilise un quantile de l'histogramme ($\frac{1}{100}$ ou $\frac{1}{1000}$) pour l'estimer.
Dans le cas contraire, il faut utiliser le fond acquis lorsqu'on retire l'objet à regarder, et le soustraire. Dans tous les cas il faut bien penser à saturer la soustraction pour ne pas être gêné par des outliers négatifs. L'utilisation d'un talon (cf. [?]) n'est pas retenue car le fond diffus n'est pas du tout une composante majoritaire du signal.

### 1.3.3 Correction du biais

Le biais doit ***a priori*** être corrigé sur l'acquisition des taux d'injection (branche de gauche) et sur l'acquisition de l'objet (branche de droite). Pour le premier cas, cela vient du fait que la calibration se fait sur un miroir plan, et que la courbure de champ va réduire la qualité d'injection au retour sur les bords (qui sont défocalisés). Ce ne serait probablement pas le cas
Sur l'objet, ou dans un milieu diffusant homogène, l'injection reste moins bonne sur les bords, et cela se traduit par un biais très similaire quant à sa forme au premier.
Dans tous les cas le biais a une symétrie quasi circulaire.
L'estimation du biais se fait en divisant l'image en $N \times N$ blocs de taille fixe, puis en estimant le biais sur chaque bloc. Pour cela il faut considérer la nature de l'objet observé. Dans le cas d'un objet homogène, le biais peut être acquis en prenant la valeur moyenne ou médiane sur le bloc. Quand il y a un objet, il faut savoir si cet objet est plus sombre ou plus claire que le biais. Dans notre cas, le biais est multiplicative, et on prend donc plutôt un opérateur de moyenne ou de médiane (par rapport à un max ou min pour un biais additif).
On obtient alors une image de taille $N \times N$ qui est utilisée avec une interpolation pour trouver la valeur du biais vue par chaque fibre. On utilise pour le moment une interpolation bilinéaire, faute de mieux. La librairie inrimage pourrait être complétée pour intégrer une interpolation en spline cubique point à point.
On peut remarquer que le biais peut être estimée plus finement avec d'autres méthodes plus subtiles qu'il faudrait étudier. Notre méthode reste cependant rapide et semble suffisante.
Une fois le biais estimé, il faut diviser l'image par son biais. Il faut penser à vérifier que le biais est bien supérieur à un pour ne pas multiplier en fait ! Dans le cas de l'image de calibration, on va

rétablir une dynamique fictive sur 1000 niveaux afin de ne pas avoir de problèmes avec la seconde division pour la calibration du taux d'injection (cf. partie 1.4.1).

## 1.4 Bloc de calibration et reconstruction

### 1.4.1 Calibration des taux d'injection

Le bloc d'estimation du flux de gauche permet d'obtenir une image des taux de calibration fibre à fibre. On peut dont l'utiliser pour rétablir une injection de 100% sur toutes les fibres. Pour cela on divise l'image de l'objet par l'image du taux d'injection, après avoir pris quelques précautions sur les valeurs de la calibration (supérieures à 1, avec suffisamment de dynamique).

### 1.4.2 Reconstruction mosaïque

La reconstruction mosaïque s'effectue en répartissant sur toute la surface de chaque fibre la valeur estimée du flux prise après l'image de calibration.

### 1.4.3 Reconstruction par RBF

A faire...

## 1.5 Résultats et discussion

Les résultats sont visibles dans la base de données d'images...

### 1.5.1 Exemple

Les figures 1.5 et 1.6 illustrent toutes les étapes du traitement de l'image. Dans ce cas de figure, le fond de l'image de calibration des fibres est estimé, ce pourquoi il n'apparaît pas sur les images. Le fond de l'objet est par contre fourni. Les images de calibration et l'objet sont toutes les deux dé-biaisées. Ce résultat a été obtenu avec les paramètres par défaut. Sur cet exemple ; on remarque que les images des biais n'ont pas la symétrie circulaire habituelle. La raison est que la majorité du biais venait en fait de la saturation du détecteur qui se traduisait par un nouveau fond additif non prévu au départ, mais qui est tout de même atténué par les traitements.

### 1.5.2 Discussion

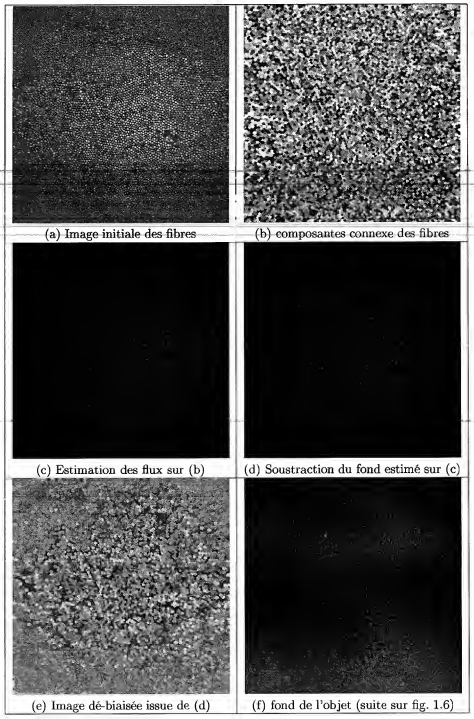
De nombreuses idées d'améliorations ont été proposée dans ce chapitre. Certaines semblent plus importantes que d'autres. Voici la proposition d'un tri de ces idées par ordre décroissant d'importance.

**Reconstruction par base de fonctions radiales :** cela permet d'obtenir une image lisse et de faire d'éventuels traitements ultérieurs.

**Supprimer le zoom sur l'image :** le zoom est utile pour diminuer l'effet de l'ouverture numérique avant la LPE. Il faut trouver l'opérateur équivalent sur l'image non zoomée. Une hypothèse à creuser est que ce serait équivalent à une ouverture avec un élément structurant en croix (?).

**Auto-contrôle de la détection des fibres :** savoir si la détection est correcte, ou si au cours des acquisition on arrive à un moment où il faut la refaire.

**Auto-contrôle de valeur du fond :** savoir sir le fond proposé est valable ou si il faut l'estimer. Dans le cas où le fond n'est pas fournit, en faut-il un ?

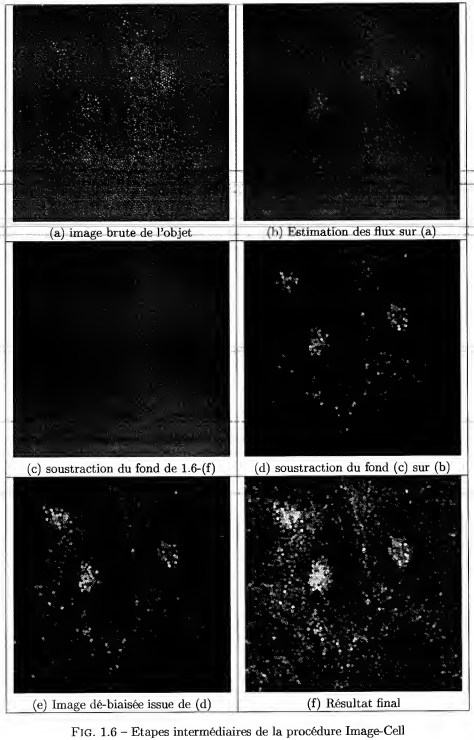


(a) Image initiale des fibres (b) composantes connexe des fibres

(c) Estimation des flux sur (b) (d) Soustraction du fond estimé sur (c)

(e) Image dé-biaisée issue de (d) (f) fond de l'objet (suite sur fig. 1.6)

FIG. 1.5 – Etapes intermédiaires de la procédure Image-Cell



(a) image brute de l'objet

(b) Estimation des flux sur (a)

(c) soustraction du fond de 1.6-(f)

(d) soustraction du fond (c) sur (b)

(e) Image dé-biaisée issue de (d)

(f) Résultat final

FIG. 1.6 – Etapes intermédiaires de la procédure Image-Cell

**Amélioration continue de la détection des fibres :** on devrait pouvoir utiliser les images des objets pour renforcer ou corriger la détection des fibres.

**Estimer le biais avec des splines :** le biais est estimé par rééchantillonnage bilinéaire qui produit un effet de blocs qui peut être gênant

On peut aussi prévoir plusieurs axes de recherche pour des fonctionnalités futures : ce sera le sujet d'un prochain mémo...

# Chapitre 2

# Programmation d'Image-Cell

## Introduction

Cette partie rassemble les principaux éléments de la librairie. La description de ces élément est globale, cette documentation venant en complément de la documentation automatique[1] décrivant toutes les classes et toutes les fonctions, et des tests unitaires décrivant l'utilisation des classes et fonctions principales.

## 2.1 Une librairie sur les graphes

La programmation des algorithmes a nécessité l'utilisation de structure de graphes. Nous avons donc recherché des librairies C++ permettant de les manipuler. Nous avons testé les librairies suivantes.

**GTL :** pour Graph Template Library (`http://www.infosun.fmi.uni-passau.de/GTL/`). Cette librairie est la plus adaptée à nos besoins et a été celle utilisée pour le prototypage dans [?]. Le problème principal de cette librairie est son prix dissuasif pour les entreprises : 5 000 Euros (licence de développement pour un seul poste). Elle était gratuite pour le stage car utilisée par une étudiante... Nous n'avons donc pas choisi de garder cette librairie pour cette raison.

**LEDA :** pour Library of Efficient Data Types and Algorithms (`http://www.mpi-sb.mpg.de/LEDA/`). Cette librairie traite de nombreux problème de type de données, dont celui des graphes. Cette librairie a cependant beaucoup évoluée vers une sorte de librairie couteau suisse qui fait beaucoup de choses, dont les interfaces graphiques, et qui redéfinit de nombreux outils maintenant standard, notamment les types STL. Il existe donc de nombreux cas de conflits potentiels avec nos softs. Le prix de cette librairie est de 1500 US$ pour un développeur, et 8600$ pour développer sur tout un site, et vendre des produits derrière. J'ai personnellement utilisé cette librairie pour ma thèse (gratuitement car pour une thèse...).

**BOOST :** une sorte de groupware qui travail sur des proposition de futurs standard C++ (`http://www.boost.org/index.htm`). Il y a en fait toute un librairie avec de nombreux thèmes. Le point central est que c'est de la programmation avancée, générique, réutilisable, versatile. C'est beau mais c'est très compliqué à utiliser malgré une documentation très propre. Le problème vient des pré-requis indispensables en programmation objet de haute voltige qui sont un peu trop complexes pour nos besoins, et qui nécessiterai un travail supplémentaire important. Cette librairie est cependant gratuite, et très bien faite. Un attrait intéressant est qu'elle n'est constituée que de fichiers .h, sans partie à compiler. Tout repose

[1] Utilisant doc++

sur une gestion complexe de templates qui ne peuvent pas être précompilée. Il n'y a donc pas de problèmes de compilation sur différentes plateformes.

Toutes ces librairies fonctionnent sous toutes les plateformes courantes, et avec tous les compilateurs courant. De plus elles sont maintenues et stables. Il existe de nombreuses autres petite libraires à droite à gauche mais qui sont loin d'approcher la qualité de celles-ci. Nous n'avons cependant pas pu en retenir une celle, soit à cause du prix et des risques liés à l'utilisation d'une grosse librairies supplémentaire, soit à cause de la complexité d'utilisation trop grande.
Nous avons donc choisi de prendre nos propres structures de données, adaptés aux traitement pour plus de rapidité à la fois de développement et d'exécution. Ces structures de données sont cependant assez souples car elle fonctionnent sur des types et des algorithmes STL qui peuvent être interchangeables assez rapidement...

## 2.2 Gestion générale des paramètres d'algorithmes

### 2.2.1 L'objet paramètre

L'objet paramètre n'existe pas encore en tant que tel, mais va bientôt voir le jour... Il est pour le moment à l'état de concept dans le même sens que la terminologie STL.
Une classe paramètre est de type suivant :

```
class AlgoParam {

  public :

  AdjGraphParam(){
    defaults();
  };

  void defaults() { // Cette fonction fixe les paramètres par défaut

  }

  /// Load parameters from a file. If any error occurs, default values are provided.
  void loadParam(const char * fname);

  /// Save parameters in a file.
  void saveParam(const char * fname);

  /// Output stream
  friend std::ostream & operator << (std::ostream &, AlgoParam &);

  /// Input stream
  friend std::istream & operator >> (std::istream &, AlgoParam &);

public: // parameters list here, with comments please !

[...]

};
```

Les fonctions loadParam et saveParam vont juste créer un flot dans le fichier dont le nom est spécifié. Ils délèguent donc le travaille aux flots `ostream` et `istream`.

Pour le moments les flots sont simples, et ne font que écrire le nom du paramètre suivie de la valeur numérique, le tout sur une ligne. Les paramètres sont encadrés par

```
ALGOPARAM_CONFIGURATION
PARAMETERS
```

et

```
END_ALGOPARAM_CONFIGURATION
```

Ce mécanisme (surement perfectible...) permet de déléguer les entrées et les sorties aux flots qui peuvent donc être imbriqués. Ainsi un objet paramètre peut contenir un autre objet paramètre sans avoir à récrire les entrées et les sorties de cet objet.
De telles classes de paramètres sont utilisés pour `ImageCell` `FiberDetector` `AdjGraph` et `FiberFlow`.

### 2.2.2 Le futur des paramètres

La première chose à faire est de créer une classe virtuelle (c'est-à-dire une interface) pour la classe de paramètres.
Enfin, dans un second temps est à l'étude l'utilisation d'XML pour interfacer ces paramètres.

## 2.3 La détection des fibres : FiberDetector et AdjGraph

Voir le test unitaire `test_fiberdetect` pour les détails d'utilisation.
L'objet FiberDetector est un objet-algorithme modulaire pour détecter les fibres. Il contient un objet spécialisé AdjGraph qui permet de faire l'étape de fusion des fibres qui est la plus compliquée.

### 2.3.1 Fonctionnement détaillé

On retrouve les étapes suivantes dans la classe :

```
void init(Inrimage *inrIn);
void preprocessing();
void watershed();
int splitBiggest ();
int mergeSmallest ();
```

L'étape d'initialisation `init` permet de fixer l'image servant de base à la détection des fibres. Puis l'étape `preprocessing` fait es premiers traitements : zoom, ouverture numérique, inversion de l'image, et quelques calculs préliminaire, comme une structure de voisinage des fibres. Ces deux étapes permettent de préparer l'image pour les traitements de segmentation.
La segmentation est réalisée par une ligne de partage des eaux (LPE) dans la fonction `watershed`. L'algorithme pourrait également interfacer un autre algorithme...
Les deux étapes suivantes sont deux étapes de post-processing de *split and merge*. `splitBiggest` va repérer les fibres trop grosses, puis les resegmenter. `mergeSmallest` permet de fusionner les fibres les plus petites. Cette dernière étape plus complexe et plus critique est elle-même encapsulée dans une classe algorithme AdjGraph expliquée dans la prochaine partie.
Ces deux étapes contiennent une dizaine de paramètres qui permettent d'ajuster la précision et la sensibilité de l'algorithme. La structure de l'algorithme permet d'éventuellement boucler ces étapes afin de faire converger l'algorithme vers une segmentation stable au sens de l'idempotence du *split and merge*, pour un jeu de paramètre donné[2].

[2] Nous n'avons pas de preuve de l'existence d'une telle convergence et il semble difficile *a priori* d'en obtenir.

### 2.3.2 L'étape de fusion avec AdjGraph

Cette partie décrit l'étape de fusion. Cette étape est importante car les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final. La complexité de cette partie nos a obligé à en faire une partie modulaire, au même titre que le `FiberDetector`. Les différentes «modules» ont été les suivants :
- type de calcul de la compacité d'une région, et de la fusion de deux régions
- construction du graphe d'adjacence dédié
- structures de données dédiée aux calculs des attributs utiles.

Cette étape était la plus longue sur le prototype de [?], et a donc fait l'objet d'optimisation. L'étape de fusion n'est réalisé que sur des critères de formes, sans lien directe nécessaire avec l'image de départ[3]. Le problème principal est le calcul des attributs utilisés : la compacité et la surface. La principale difficulté est liée à ce que le graphe va évoluer par fusion de régions, et que l'on cherche à mettre à jour ces attributs ; on veut également tester les fusions possibles et donc pré-calculer les attributs possibles.
Pour cela, on distingue les frontières extérieures et intérieures d'un objet comme indiqué sur la figure 2.1. On divise la frontière d'un objet en fonction de l'adjacence. La frontière de A est égale à la somme des frontière intérieures de A avec tous ses voisins. La frontière intérieure de A commune avec F est égale à la frontière extérieure de F commune avec A. Cette dernière propriété permet de ne parler que de frontière intérieure.

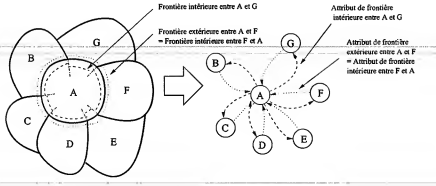



FIG. 2.1 – Les frontières extérieurs et intérieures d'un objet. La frontière intérieure de A commune avec F est égale à la frontière extérieure de F commune avec A. Ces informations sont portés par des arcs du graphe d'adjacence.

Ces informations sont portés par des arcs du graphe d'adjacence. Un arc allant de A vers B va porter l'information de frontière commue que A détient avec B.
On peut formaliser ces relations ainsi. Soit $G = (N, E)$ un graphe avec $N$ l'ensemble de ses noeuds et $E \subseteq N \times N$ l'ensemble de ses arcs. G est un digraphe[4] où deux nœuds n'ont soit pas d'arcs en commun, soit deux arcs antisymétriques. $E$ est donc un ensemble de paires ordonnées. Soit $s : N \to \mathbb{R}^+$ une fonction permettant d'obtenir la surface. Soit $f : E \to \mathbb{R}^+$ une fonction permettant d'obtenir la frontière intérieure entre les deux nœ désigné par l'arc. Enfin soit $c : N \to \mathbb{R}^+$ la fonction permettant d'obtenir la compacité d'une région. On note $\mathcal{V}(n)$, $n \in N$ l'ensemble des voisins de $n$. La compacité peut alors s'écrire :

$$c(n) = \frac{\left(\sum_{m \in \mathcal{V}(n)} f((n,m))\right)^2}{s(n)}. \tag{2.1}$$

[3]Ce qui est en fait une extension (lourde !) possible
[4]un graphe où tous les arcs sont dirigés

Pour tester la compacité de la fusion éventuelle de $n$ et $m$, on note la nouvelle compacité $c(n \cup m)$ :

$$c(n \cup m) = \frac{\left(\sum_{o \in \mathcal{V}(n)} f((n,o)) + \sum_{o \in \mathcal{V}(m)} f((m,o)) - f((n,m)) - f((m,n))\right)^2}{s(n) + s(m)}. \quad (2.2)$$

Et si la fusion de $m$ avec $n$ est effectuée, on modifie les fonctions (on suppose que $m$ est rajoutée à $n$ ) :

$$s(n) \leftarrow s(n) + s(m) \quad (2.3)$$
$$f(n,o) \leftarrow f(n,o) \text{ pour } o \in \mathcal{V}(n) \text{ et } o \notin \mathcal{V}(m) \quad (2.4)$$
$$f(n,o) \leftarrow f(n,o) + f(m,o) \text{ pour } o \in \mathcal{V}(n) \cap \mathcal{V}(m). \quad (2.5)$$

On peut ainsi ne travailler que sur les graphes, sans avoir à revenir à la segmentation initiale pour recalculer les attributs lors d'une fusion. Cela nous a permis de tester beaucoup plus de fusion, voire éventuellement de tester toutes les fusions possibles.

## 2.4 Structures de données associées aux fibres : FiberStructure et FiberInfo

Voir le test unitaire `test_fiberInfo` pour les détails d'utilisation.

### 2.4.1 Lien entre la structure des fibres et les informations sur chaque fibre

La structure de donnée est spécifique au problème qui nous intéresse. Les informations qui nous intéressent sont des informations attachées aux fibres seulement, les fibres étant en fait liées à la structure détectée. Les deux types d'informations sont donc étroitement liées, les informations sur les fibres ne pouvant exister indépendamment de la structure des fibres.

#### Modèle sujet-observateur

Nous avons donc choisi de les relier sous le modèle de l'observateur des design patterns [?]. Nous avons donc défini ce mécanisme dans le fichier `Observer.h`. Le principe est le suivant :

> «Définit une interdépendance un à plusieurs, de façon telle que, quand un objet change d'état, tous ceux qui en dépendent en soient notifiés et automatiquement mis à jour.»

Ainsi on distingue le sujet, qui pour nous est la structure des fibres détenue par la classe `FiberStructure` ; et les observateurs qui sont les informations que l'on peut attacher aux fibres (classe `FiberInfo<T>`).
On a ajouté un comportement supplémentaire : un objet `FiberInfo<T>` ne peut exister sans un `FiberStructure`, ce pourquoi il n'existe qu'un constructeur qui prend en argument la `FiberStructure`. L'objet `FiberInfo<T>` va d'ailleurs s'initialiser tout seul en regardant la structure à laquelle il est attaché.
Quand la structure vient à changer, tous les `FiberInfo<T>` en sont informés si besoin, et ceux-ci vont alors se réinitialiser automatiquement. Quand la structure est détruite, les objets d'information sont informés également qu'ils ne sont plus valides.

#### Structure codée comme un graphe : FiberStructure

La classe fiberstructure se construit à partir de l'image des composantes connexes. L'objet va alors garder en interne une copie de cette image de référence. Elle contient en interne la structure de graphe associée alors aux composantes connexes. Elle contient également d'autres informations, comme les barycentres des régions, en coordonnées pixel, les surface des régions ...

Toutes ces informations sont accessibles depuis l'interface de la classe, mais ne sont pas modifiables, car elles dépendent de la structure même.

#### La classe d'information : FiberInfo

La classe `FiberInfo<T>` est une classe template car les informations peuvent être de nature très différentes. Elle contient un vecteur STL d'information dont la taille est gérée automatiquement en fonction de la structure associée. Ce vecteur est un membre *public* afin d'avoir des accès rapides aux informations. La classe sert juste de décorateur[5] pour pouvoir gérer les dépendances avec la structure des fibres.
Quand le type du template est numérique, il existe des fonctions pour additionner, soustraire, diviser et multiplier deux `FiberInfo<T>`, ou bien un `FiberInfo<T>` avec un élément de `T` : `addFI`, `subFI`, `divFI`, `mulFI`.

### 2.4.2 Transformation d'images en FiberInfo et réciproquement

Voir le test unitaire `test_fiberProcessing` pour les détails d'utilisation.
Une image peut être transformée en `FiberInfo<float>` avec la fonction d'estimation simple des flux :
`int flowEstimation (Inrimage* image, FiberInfo<float> & flow, bool bias = false)`.
Il faut au préalable initialiser de `FiberInfo<float> flow` avec la structure, ainsi l'algorithme connaît via `flow` l'image des composantes connexes à utiliser.
La transformation réciproque est la reconstruction d'image, pour laquelle il n'existe pour le moment que la méthode de mosaïquage :
`int imageMosaic (Inrimage* image, FiberInfo<float> & flow)`
Le principe est similaire à `flowEstimation`. La structure est passée via le `flow`.

## 2.5 Soustraction du fond et biais : FiberFlow

Voir le test unitaire `test_fiberFlow` pour les détails d'utilisation.

### 2.5.1 Fonctionnement général

La classe FiberFlow implémente le bloc d'estimation des flux de la partie 1.1, sauf la partie de transformation de l'image en `FiberInfo`. La classe comporte deux méthodes principales : `setBackground` et `setImage`, et son comportement est paramétré avec la classe de paramètre `FiberFlowParam`. La classe permet de soustraire le fond qui est donnée, ou bien il peut l'estimer. La classe sert aussi à calculer le biais dans l'image, après soustraction du fond.

### 2.5.2 Le calcul du fond

Voir le test unitaire `test_fiberHisto` pour les détails d'utilisation.
Pour calculer le fond, on calcul un quantile de l'histogramme des fibres. Pour cela il existe une classe `FiberHisto<T>` qui prend un `FiberInfo<T>` en constructeur, et qui permet de faire des calculs sur l'histogramme calculé à partir de l'information des fibres. Ces calculs sont plus rapides que sur l'image elle-même.

[5] Attention, pas au sens du Design Pattern décorateur [?] !

### 2.5.3 Le calcul du biais

Voir le test unitaire `test_fiberProcessing` pour les détails d'utilisation.
Le calcul du biais n'est en fait pas calculé sur l'image, mais encore une fois sur l'information des fibres directement. C'est encore un cas où on utilise la structure et l'information des fibres. Ce calcul est fait dans une fonction :

```
int biasEstimate  ( FiberInfo<float> & flowIn,
                    FiberInfo<float> & bias,
                    int nbBlocks,
                    biasEstimatorType bet=BET_MEAN)
```

L'estimateur du biais est pour le moment à choisir entre la moyenne (`BET_MEAN`) et la médiane (`BET_MEDIAN`).
On divise l'espace total des fibres en `nbBlocks` blocs de taille égale. On parcourt ensuite l'ensemble des blocs et on trie les fibres en fonctions du bloc dans lequel elles tombent. Puis, pour chaque bloc, on applique l'opérateur choisi d'estimation du biais.
On calcule le biais ensuite en interpolant pour chaque fibre uniquement (et non pour toute l'image) les valeurs des blocs les plus proches. On utilise pour le moment une interpolation bilinéaire, car c'est la seule disponible avec Inrimage. On fait une proposition pour avoir une interpolation spline cubique.

## 2.6 Améliorations

- La classe de paramètres est à optimiser et à programmer plus proprement.
- Le biais devrait être interpolé avec une spline cubique
- Une librairie de graphe serait tout de même la bienvenue. Cela permettrait d'avoir une structure commune entre `FiberDetector`, `AdjGraph`, et `FiberStructure`, ce qui n'est actuellement pas exactement le cas.

# Chapitre 3

# Utilisation d'Image-Cell

C'est une partie délicate car elle doit contenir le savoir-faire acquis pendant le prototypage, le développement et les nombreux tests.
Les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final.

## 3.1 Utilisation de la paramétrisation

Il faut ouvrir une fenêtre d'un éditeur de texte (notepad, emacs, wordpad...) avec le fichier imagecell.txt. Ce fichier est relu à chaque utilisation. La liste des paramètre, ainsi que des résumés d'indications sont données en annexe A.

## 3.2 Paramétrage de FiberDetector

### 3.2.1 Prétraitements

Dans presque tous les cas, il faut faire le zoom sur l'image pour que l'algorithme fonctionne correctement.

### 3.2.2 Paramétrage de AdjGraph

Les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final.

## 3.3 Paramétrage du FiberFlow pour la calibration

## 3.4 Paramétrage du FiberFlow pour l'image à reconstruire

## 3.5 Paramétrage d'ImageCell : les reste des paramètres

# Annexe A

# Paramètres de Image-Cell

Les paramètres internes d'Image-Cell sont modifiables via un fichier de paramètres présent dans le répertoire source du lancement de l'application, au même endroit que tomoscope.cfg. Le fichier des paramètres `imagecell.cfg` a la forme suivante :

| Ligne du fichier | Défaut | Possibles | Remarques |
|---|---|---|---|
| IMAGECELLPARAM_CONFIGURATION | | | Paramètres de l'algorithme entier |
| PARAMETERS | | | - |
| FIBERDETECTORPARAM_CONFIGURATION | | | Paramètres de l'algorithme entier |
| PARAMETERS | | | - |
| _doZoom_ | 1 | $\{0,1\}$ | Travail sur l'image doublée (plus précis) |
| _doDiffusion_ | 0 | $\{0,1\}$ | Option non pris en compte pour le moment. A régler. |
| _hDomeWatershed_ | 1 | $\{1,..,10\}$ | Hauteur des maxima sélectionnés pour initialiser les fibres. |
| _oversizedFactor_ | 1.8 | $[1;2]$ | Taille normalisée (par rapport à la moyenne) au delà de laquelle une fibre va être divisée. Ce critère est utilisé avec le critère suivant sur la taille (ET logique) |
| _overSizedNeighLimit_ | 6 | $\{5,6,7\}$ | Nombre de voisins au delà duquel une fibre va être divisée. Ce critère est utilisé avec le critère précédent sur la taille (ET logique) |
| _overSizedResegmentMethod_ | 1 | $\{0,1\}$ | 0 : resegmenter dans une fonction distance ; 1 : resegmenter dans l'image originale sans prétraitement |
| ADJGRAPHPARAM_CONFIGURATION | | | Paramètres pour la fusion des fibres |
| PARAMETERS | | | - |
| _underSizedFactor_ | 0.6 | $[0.1;1]$ | Taille normalisée (par rapport à la moyenne) en deçà de laquelle une fibre va être une candidate pour la fusion. Ce critère est utilisé avec le critère suivant sur la taille (ET logique) |
| _underSizedMendatoryMergeSlope_ | $-0.22$ | $[?;?]$ | Ce paramètre et le suivant déterminent une droite dans l'espace (nombre de voisins)x(taille normalisée) délimitant les fibres qui doivent obligatoirement être fusionnées. *Mieux vaut ne pas y toucher sans faire de calcul précis !* La droite par défaut passe par les points $(6;1/3)$ et $(3,1)$. Augmenter la pente revient à s'occuper plus des grosses fibres avec peu de voisins. |
| _underSizedMendatoryMergeOffset_ | 1.667 | $[?;?]$ | Voir remarque précédente. Ce paramètre est l'ordonnée à l'origine (axe des ordonnées=tailles normalisées). L'augmenter revient à obligatoirement fusionner plus de fibres. |
| _underSizedFilterSize_ | 1 | $\{0,1\}$ | Doit-on filter les fusions possibles en fonction de la taille totale de la fusion ? |
| _underSizedFilterSizeMax_ | 1.17741 | $[1;3]$ | Si oui, interdire les fusions donnant une taille de fibre normalisée supérieure à |
| _underSizedFilterCompact_ | 0 | $\{0,1\}$ | Filtrage des fibres fusionnées en fonction de la compacité |
| _underSizedFilterCompactMax_ | 2 | $[1;10]$ | Facteur multiplicatif de l'écart-type |

| Ligne du fichier | Défaut | Possibles | Remarques |
|---|---|---|---|
| _underSizedFilterMoreCompact_ | 0 | {0,1} | Ne garde que les fusions qui rendent les fibres fusionnées plus compacte que la fibre voisine seule. |
| END_ADJGRAPHPARAM_CONFIGURATION | | | |
| END_FIBERDETECTORPARAM_CONFIGURATION | | | |
| FIBERFLOWPARAM_CONFIGURATION | | | Paramètres d'estimation du flux pour l'image de calibration |
| PARAMETERS | | | - |
| _backgroundMode_ | 1 | {0,1,2} | 0 : pas de soustraction du fond ; 1 : soustraction avec le fond fourni, sinon 2 ; 2 : estimation du fond sur l'histogramme |
| _backgroundEstimationCut_ | 0.01 | [0;0.1] | Quantile de l'histogramme de l'image pour l'estimation du fond |
| _biasCorrection_ | 1 | {0,1} | Correction du biais, après soustarction du fond |
| _biasCorrectionNbBlocks_ | 16 | {2,..,64} | Si correction du biais, nombre de blocks en X et en Y pour son estimation |
| _biasCorrectionOperator_ | 1 | {0,1} | Opérateur pour la correction du biais : 0 : moyenne ; 1 : médiane |
| END_FIBERFLOWPARAM_CONFIGURATION | | | |
| FIBERFLOWPARAM_CONFIGURATION | | | Paramètres d'estimation du flux pour l'image à reconstruire |
| PARAMETERS | | | - |
| _backgroundMode_ | 1 | {0,1,2} | 0 : pas de soustraction du fond ; 1 : soustraction avec le fond fourni, sinon 2 ; 2 : estimation du fond sur l'histogramme |
| _backgroundEstimationCut_ | 0.01 | [0;0.1] | Quantile de l'histogramme de l'image pour l'estimation du fond |
| _biasCorrection_ | 1 | {0,1} | Correction du biais, après soustarction du fond |
| _biasCorrectionNbBlocks_ | 4 | {2,..,64} | Si correction du biais, nombre de blocks en X et en Y pour son estimation |
| _biasCorrectionOperator_ | 1 | {0,1} | Opérateur pour la correction du biais : 0 : moyenne, 1 : médiane |
| END_FIBERFLOWPARAM_CONFIGURATION | | | |
| _tempoFilterTypeFib_ | 3 | {0,..,4} | Type du filtre temporel pour l'estimation des fibres :0 : moyenne ;1 : moyenne coupée ; 2 : médianne ; 3 : max ; 4 : min |
| _calibration_ | 1 | {0,1} | Effectuer la calibration |
| END_IMAGECELLPARAM_CONFIGURATION | | | |

# Bibliographie

**Diligence Document 6**

# Manuel de référence du module ImageCell

Auteur : Aymeric Perchant

Date : 2002-03-12

**Diffusion :** interne

**Section :** informatique, image

**Sujet :** Manuel de référence de la partie libMKTProcessing pour imageCell. Ce manuel contient les informations de traitement d'images, de programmation C++, et d'utilisation.

**Version :** *Revision* : 1.2

**Référence du document :** `$Id: ReferenceImageCell.tex,v 1.2 2002-03-12 10:05:01 aymeric Exp $`

## Introduction

Ce document rassemble les différentes informations de référence pour le module principal intitulé Image-Cell. Ce module contient les parties suivantes de traitement d'images :

- détéction des fibres,
- estimation des flux revenant des fibres,
- corrections des défauts de l'appareil (biais, fond, ...),
- calibration de l'appareil,
- reconstruction d'images.

Ces différentes parties seront abordées en détail dans chaque partie pour expliquer les algorithmes et les paramètres (chapitre 1), l'implantation de ceux-ci (2) et leur utilisation pratique (3).

### Corrections

- rajout d'un filtrage passe bas, mars 2002.

# Table des matières

# Chapitre 1

# Traitement d'images dans Image-Cell

## Introduction

Cette partie décrit les algorithmes en commençant par la description générale, jusqu'aux boites élémentaires.

## 1.1 Schéma global

La figure 1.1 représente le schéma global du traitement. On peut distinguer quatre blocs, dont deux sont identiques, à une paramétrisation près. Il existe donc trois groupes de traitements qui sont les suivants :

**Détection des fibres :** ces traitements permettent de détecter et d'isoler chaque fibre sur une image, ainsi que d'analyse la structure d'agencement des fibres du guide d'images ;

**Estimation des flux :** une image brute de taille $640 \times 640 \times 20$ (largeur, hauteur, nombre d'images temporellement )contient 8 méga pixels qui représentent l'information vue par 10000 ou 30000 fibres. Ce bloc permet d'isoler l'information effectivement vue par chaque fibre ;

**Calibration et reconstruction :** La connaissance de l'information vue par chaque fibre est ensuite traitée pour être reconstruite sous la forme d'une image débarrassée des défauts de l'appareil.

Chacun de ces trois blocs sont maintenant détaillés. Le prototypage des algorithmes décrits ici est détaillé dans le rapport de stage de Sandra Marti [?] ; nous renvoyons le lecteur à cette référence pour plus de renseignements sur la démarche de développement des algorithmes, notamment celui de détection des fibres.
Le bloc de gauche d'estimation des flux permet de mesurer le flux sur une image qui représente un objet aux propriétés constantes dans l'espace (un milieu diffusant homogène ou un objet presque homogène en mouvement aléatoire, un miroir). La sortie de ce bloc est donc une image des taux d'injection dans chaque fibre, et qui en prend en compte l'intégralité de la chaîne.
Le bloc de droite d'estimation des flux permet de l'estimer pour un objet à observer, et à travers l'appareil. Le dernier bloc de calibration est une division de l'image des taux d'injection par l'image de l'objet observé. Cette opération permet de compenser les mauvaises injections dans certaines fibres. Puis l'image est reconstruite.

## 1.2 Bloc de détection des fibres

La détection des fibres s'organise autour de quatre traitements successifs :
– prétraitements,

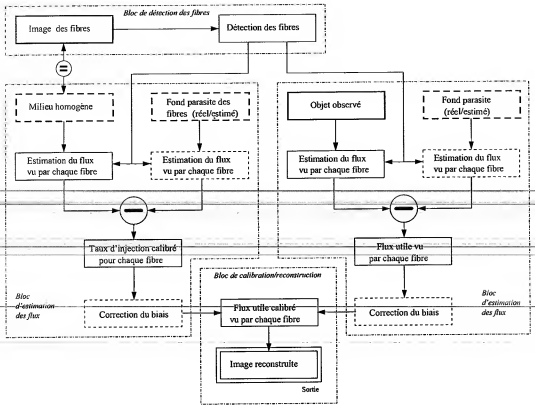



FIG. 1.1 – Schéma global du traitement. Les cadres gras représentent les entrées. Les cadres pointillés représentent les données ou les traitements facultatifs (biais, soustraction du fond...).

- ligne de partage des eaux (LPE) : la première segmentation par région,
- séparation des fibres trop grosses (split),
- fusion des fibres trop petites (merge).

Les deux dernières étapes peuvent être bouclées ; néanmoins une analyse de performance et de convergence doit alors être menée. Nous n'aborderons pas ce sujet pour le moment.

### 1.2.1 Prétraitements

Les prétraitements réalisés sont les suivants.

**Diffusion anisotrope :** le but est de lisser l'image dans les zones plates, c'est à dire les zones inter fibres. Le prototype fonctionnait mais n'utilisait pas la librairie inrimage, mais une autre librairie (celle utilisée par Sandra, issue de l'ENST). Le réglage de la diffusion est laissée à plus tard, lors de la prochaine release de la libInrimage qui contiendra de la documentation à ce sujet ! Paramètre `[_doDiffusion_]`.

**Interpolation $\times 2$ au plus proche voisin :** on cherche à simuler des éléments structurant de morphologie mathématique avec un rayon inférieur à un. L'image est doublée pour que l'ouverture qui suit ne touche pas aux maxima isolés, mais seulement ceux qui sont 8-connexe, mais non 4 connexe (voisins par une diagonale). L'intérêt est de faire une sélection des maxima éliminé par l'ouverture. Paramètre `[_doZoom_]`.

**Ouverture numérique :** on cherche à éliminer les maxima parasites situés sur les fibres. C'est un prétraitement classique de la LPE, mais éventuellement modifié par le comportement du doublage de l'image expliqué précédemment.

**Inversion de l'image :** la LPE fonctionne à partir de minima, on inverse donc l'image pour transformer les minima en maxima.

L'opération d'interpolation et d'ouverture qui suit devrait pouvoir être remplacé par un nouvel opérateur de morphologie mathématique qu'il faudrait définir. Cela permettrait un gain important, puisque toutes les opérations suivantes se feraient sur une image de taille inférieure.

### 1.2.2 Premier watershed

Le watershed de la libInrimage nécessite la définition de marqueurs dans l'images. Nous obtenons les marqueurs comme minima de l'image prétraitée avec de h-dômes [7]. Les h-domes pour les minima sont définis pour un entier $h$, une fonction $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ représentant l'image concernée :

$$g = E_f^\infty(f + h) - f, \tag{1.1}$$

avec $E_a^\infty(b)$ la reconstruction géodésique par érosion de $b$ dans $a$. $g$ désigne ici l'ensemble des h-domes, que l'on seuil à 1 afin de tous les récupérer. On obtient ainsi les minima de profondeur au moins égale à $h$. Ce paramètre $h$ est donné par `[_hDomeWatershed_]`.
Le résultat de cette LPE est une image de composantes connexes représentant chaque fibre détectée. Afin de résoudre bon nombre de problèmes de bord on colle les composantes touchant le bord au bord lui-même. Ces composantes sont considérées comme faisant partie du bord, et du fond et sont mise à la composante connexe d'indice 0, qui est l'indice réservé au fond, et au bord.
Enfin, les composantes connexes sont réorganisées pour ne pas présenter de trous, et sont triés par ordre décroissant de taille.

### 1.2.3 Split

Les prétraitements, et notamment l'ouverture numérique servent à limiter le défaut principal de la LPE qui est de sursegmenter. En faisant cela, on a tendance à augmenter le nombre de segments qui vont englober plusieurs objets (ici, des fibres). Le premier défaut que l'on corrige est la sous-segmentation inévitable au traitement, mais surtout rajoutée par ces prétraitements. Pour cela on effectue les traitements suivants.

**Sélection des segments suspectés de sous-segmentation :** on trouve les fibres dont la taille est supérieure à un seuil fixé en fonction de la moyenne des tailles normalisées (paramètre `[_oversizedFactor_]`), et en fonction du nombre de voisins (paramètre `[_overSizedNeighLimit_]`).

**Re-segmentation de ceux-ci :** Les segments sélectionnés sont isolées et re-segmenter soit sur l'image initiale sans prétraitement, soit sur l'image de carte de distance à l'intérieur de ces régions (paramètre `[_overSizedResegmentMethod_]`). Dans le premier cas, on suppose que les défauts majoritaires proviennent des prétraitement, dans le second cas on suppose qu'ils étaient là avant, et que l'on souhaite séparer les segment aux endroits de rétrécissement (forme de la cacahuète). La LPE est effectuée avec les même paramètres que la première.

### 1.2.4 Merge

Puis on cherche à corriger le défaut le plus classique (pour une LPE) : la sursegmentation. On cherche donc à fusionner les segments. Pour cela on effectue les opérations suivantes :
- présélection des *candidats à la fusion*,
- parmi ceux-ci trier ceux qui seront obligatoirement fusionnés,
- pour les autres, éliminer les fusions impossibles
  - répertorier les fusions possibles

- éliminer les fusions donnant de mauvais résultats
- si il ne reste pas de fusion après filtrage, retirer la fibres des fibres à fusionner

– pour toutes les fibres restantes, prendre la meilleur fusion au sens de la compacité des segments fusionnés.

Les points importants sont détaillés ci-après.

### Remarques sur la compacité

La compacité utilisée est le critère le plus simple existant. Pour un objet de périmètre $P$ et de surface $S$, le critère est $\frac{P^2}{S}$. Le périmètre est pris comme étant la frontière de l'objet 4-connexe, et donc est défini comme tous les pixels de l'objet ayant un voisin 8-connexe avec le fond (défini ici comme étant tout le reste de l'image. La surface est simplement l'objet lui-même.

Cependant, ce critère simple à calculer n'est pas invariant avec l'échelle de l'objet. On peut montrer que pour des formes simple, le critère est croissant avec la taille de l'objet au lieu d'être invariant. Cette croissante n'est cependant qu'en moyenne en $\frac{1}{\pi r}$, avec n la diamètre de l'objet (ici, pour une boule en 4-connexité). Ce critère reste donc tout de même assez raisonnable, mais plus délicat à manier dans le cas de la comparaison de la compacité d'objets de taille différentes.

Il faut remarquer ici que les premiers essais sont réalisés sur une image déformée par les distortions, et qu'il faudrait, pour être plus précis soit corriger l'image avant, soit tenir compte de l'anisotropie non-stationnaire spatialement : on a tendance à surestimer à la fois les frontières et la surface sur les bords de l'image.

### Sélection des fibres détectées trop petites

La sélection s'effectue en deux étapes. On commence par trouver les segments candidats à la fusion avec un seuil sur la taille. On utilise le paramètre `[_underSizedFactor_]` qui est un facteur multiplicatif de la moyenne, et est donc inférieur à 1 en général. La seconde étape est de trouver parmi ces segments lesquels doivent obligatoirement être fusionnés, et lesquels doivent être filtrés.

### Les segments obligatoirement fusionnés

La sélection des segments (ou fibres) obligatoirement fusionnés s'effectue dans l'espace (nombre de voisins) $\times$ (taille normalisée). La taille est normalisée par rapport à la moyenne (on divise toutes les tailles par la moyenne des tailles). Le nombre de voisins est compté avec la 8-connexité.

Les figures 1.2 et 1.3 représentent l'histogramme conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. On remarque qu'il existe un axe principal qui est centré sur la droite passant par des points d'intérêts particuliers $(6, 1)$, $(8, 2)$, et $(5, .5)$. Ces points peuvent s'explique géométriquement, comme indiqué sur la figure 1.4

Cet axe est donc un axe naturel de la structure hexagonale des fibres dans cet espace. En s'orientant sur cet axe, les petites fibres se situent vers un nombre de voisins faibles et une taille faible. On détermine une frontière linéaire pour sélectionner les fibres détectées réellement trop petites. Après quelques tests sur des images réelles de la base de données «Novembre 2001» et en fonction des résultats préliminaires de [?], on a choisi une droite passant par les points $(6; \frac{1}{9})$ $(3; 1)$. Les paramètres de cette droites sont données sous la forme $y = mx + p$, avec $m$ la pente `[_underSizedMendatoryMergeSlope_]` et $p$ l'ordonnée à l'origine `[_underSizedMendatoryMergeOffset_]`.

Tout point en dessous de la droite sera fusionné obligatoirement, et ne va donc pas passer par les filtres décrits dans la section suivante. Les autres fibres sont par contre filtrée comme suit.

### Filtrage des fusions possibles

On entend par filtrage ici un procédé qui retire certaines fusions. Il existe trois filtres possibles.

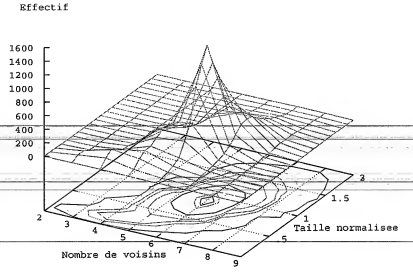



FIG. 1.2 – Vue 3d de l'histogramme 2d conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. Des lignes de niveaux sont projetées sur la fond du graphique : les lignes sont centrées autour de 6 voisins et de la moyenne de la taille des fibres.

**Filtrage sur la taille de la fusion :** (paramètre `[_underSizedFilterSize_]`) on élimine les fusions dont la taille totale dépasse un seuil fixé par le paramètre `[_underSizedFilterSizeMax_]`, qui est pris comme facteur multiplicatif de l'écart type ajouté à la taille moyenne.

**Filtrage sur la compacité totale :** (paramètre `[_underSizedFilterCompact_]`) on élimine les fusions dont la compacité totale dépasse un seuil fixé par le paramètre `[_underSizedFilterCompactMax_]`, qui est pris comme facteur multiplicatif de l'écart type ajouté à la taille moyenne.

**Filtrage sur le changement de compacité :** (paramètre `[_underSizedFilterMoreCompact_]`) on élimine les fusions qui augmentent la compacité de l'objet avec lequel on veut fusionner.

Dans les paramètres par défaut, on n'utilise pour le moment que le premier filtre qui semble suffisant dans la plupart des cas. Les autres filtres semblent trop restrictifs et doivent être encore étudiés.

## 1.3 Bloc d'estimation des flux

Le bloc d'estimation des flux comprend plusieurs sous-parties :

**estimation du flux vu par chaque fibre :** c'est ce que l'on peu récupérer d'une image à l'aide de la détections es fibres ;

**estimation du fond de l'image :** ici le fond désigne les réflexions parasites ou (*ou* inclusif !) l'offset du à l'électronique et au détecteur ;

**soustraction du fond :** on retire le fond à l'image ;

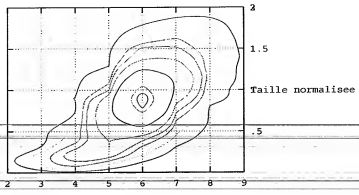



FIG. 1.3 – Lignes de niveau de l'histogramme 2d conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. La forme 3d de cet histogramme est donnée sur la figure 1.2

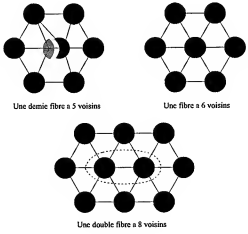



FIG. 1.4 – Variation du nombre de voisins en fonction de la taille du segment détecté comme étant une fibre

**correction du biais :** les images sont en général biaisés, c'est à dire qu'il existe un fond lentement variable au lieu d'être constant.

Le bloc de soustraction du fond est optionnel, ainsi que celui de la correction du biais.

### 1.3.1 Estimation du flux vu par chaque fibre

Le rapport [?] décrit une méthode pour *rattraper* les problèmes de saturation de l'image. Ces problème vont disparaître avec l'arrivée de l'électronique mi mai 2002. Nous avons donc décidé de ne pas continuer dans cette voie pour le moment. Ici nous avons simplement choisi d'estimer le flux avec la moyenne sur la fibre.
Plusieurs améliorations sont envisageables :

- tenir compte du bruit poissonien dans l'estimateur du max avec la moyenne,
- tenir compte de la disparité de forme de fibres du guide d'image ; on peut constater que les toutes les fibres ne sont pas identiques et que l'estimateur du flux devrait s'adapter à la forme exacte de la fibre,
- vérifier les problèmes possibles dues à la zone inter-fibre, en fonction de la tache focale sur l'entrée du guide,
- et plus généralement récupérer le modèle complet d'injection et de retour du spot décrit dans [?].

Ces améliorations peuvent être étudiées, mais ne sont pas prioritaires pour le moment.

### 1.3.2 Estimation ou utilisation du fond, puis soustraction

Le fond peut avoir plusieurs sources ; celles-ci sont décrite dans les mémos [?, ?, ?]. En résumé, nous appelons fond ici soit les réflexions parasites sur les optiques, et donc y compris sur la sortie du guide d'images, mais également l'offset due à la chaîne de numérisation.
Si l'offset est dominant sur l'image, on ne peut pas obtenir le fond simplement en retirant l'image, car l'offset dépend du contenu, et n'est donc plus le même. Dans ce cas, on utilise un quantile de l'histogramme ($\frac{1}{100}$ ou $\frac{1}{1000}$) pour l'estimer.
Dans le cas contraire, il faut utiliser le fond acquis lorsqu'on retire l'objet à regarder, et le soustraire. Dans tous les cas il faut bien penser à saturer la soustraction pour ne pas être gêné par des outliers négatifs. L'utilisation d'un talon (cf. [?]) n'est pas retenue car le fond diffus n'est pas du tout une composante majoritaire du signal.

### 1.3.3 Correction du biais

Le biais doit *a priori* être corrigé sur l'acquisition des taux d'injection (branche de gauche) et sur l'acquisition de l'objet (branche de droite). Pour le premier cas, cela vient du fait que la calibration se fait sur un miroir plan, et que la courbure de champ va réduire la qualité d'injection au retour sur les bords (qui sont défocalisés). Ce ne serait probablement pas le cas
Sur l'objet, ou dans un milieu diffusant homogène, l'injection reste moins bonne sur les bords, et cela se traduit par un biais très similaire quant à sa forme au premier.
Dans tous les cas le biais a une symétrie quasi circulaire.
L'estimation du biais se fait en divisant l'image en $N \times N$ blocs de taille fixe, puis en estimant le biais sur chaque bloc. Pour cela il faut considérer la nature de l'objet observé. Dans le cas d'un objet homogène, le biais peut être acquis en prenant la valeur moyenne ou médiane sur le bloc. Quand il y a un objet, il faut savoir si cet objet est plus sombre ou plus claire que le biais. Dans notre cas, le biais est multiplicative, et on prend donc plutôt un opérateur de moyenne ou de médiane (par rapport à un max ou min pour un biais additif).
On obtient alors une image de taille $N \times N$ qui est utilisée avec une interpolation pour trouver la valeur du biais vue par chaque fibre. On utilise pour le moment une interpolation bilinéaire, faute de mieux. La librairie inrimage pourrait être complétée pour intégrer une interpolation en spline cubique point à point.
On peut remarquer que le biais peut être estimée plus finement avec d'autres méthodes plus subtiles qu'il faudrait étudier. Notre méthode reste cependant rapide et semble suffisante.
Une fois le biais estimé, il faut diviser l'image par son biais. Il faut penser à vérifier que le biais est bien supérieur à un pour ne pas multiplier en fait ! Dans le cas de l'image de calibration, on va

rétablir une dynamique fictive sur 1000 niveaux afin de ne pas avoir de problèmes avec la seconde division pour la calibration du taux d'injection (cf. partie 1.4.1).

## 1.4 Bloc de calibration et reconstruction

### 1.4.1 Calibration des taux d'injection

Le bloc d'estimation du flux de gauche permet d'obtenir une image des taux de calibration fibre à fibre. On peut dont l'utiliser pour rétablir une injection de 100% sur toutes les fibres. Pour cela on divise l'image de l'objet par l'image du taux d'injection, après avoir pris quelques précautions sur les valeurs de la calibration (supérieures à 1, avec suffisamment de dynamique).

### 1.4.2 Reconstruction mosaïque

La reconstruction mosaïque s'effectue en répartissant sur toute la surface de chaque fibre la valeur estimée du flux prise après l'image de calibration.
Afin de donner un aspect plus lisse, on a rajouté un filtrage récursif passe bas de type Dériche. Le paramètre `[_outputSigmaFilter_]` permet de régler l'écart-type. Noter que la complexité de l'algorithme est indépendante de l'écart type de la gaussiène.

### 1.4.3 Reconstruction par RBF

A faire...

## 1.5 Résultats et discussion

Les résultats sont visibles dans la base de données d'images...

### 1.5.1 Exemple

Les figures 1.5 et 1.6 illustrent toutes les étapes du traitement de l'image. Dans ce cas de figure, le fond de l'image de calibration des fibres est estimé, ce pourquoi il n'apparaît pas sur les images. Le fond de l'objet est par contre fourni. Les images de calibration et l'objet sont toutes les deux dé-biaisées. Ce résultat a été obtenu avec les paramètres par défaut. Sur cet exemple ; on remarque que les images des biais n'ont pas la symétrie circulaire habituelle. La raison est que la majorité du biais venait en fait de la saturation du détecteur qui se traduisait par un nouveau fond additif non prévu au départ, mais qui est tout de même atténué par les traitements.

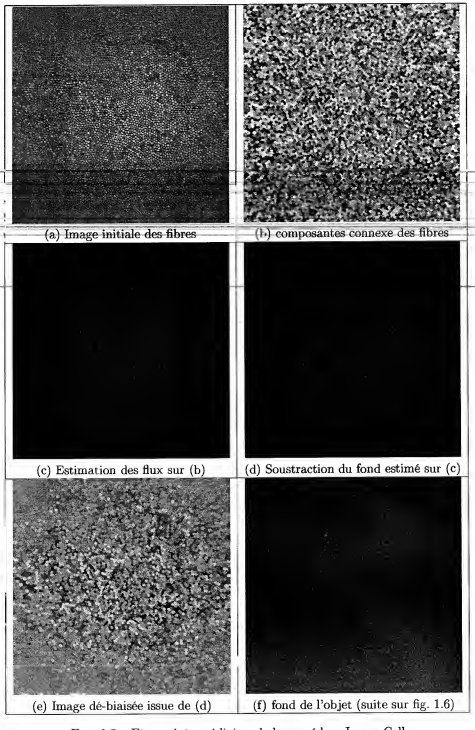
### 1.5.2 Discussion

De nombreuses idées d'améliorations ont été proposée dans ce chapitre. Certaines semblent plus importantes que d'autres. Voici la proposition d'un tri de ces idées par ordre décroissant d'importance.

**Reconstruction par base de fonctions radiales :** cela permet d'obtenir une image lisse et de faire d'éventuels traitements ultérieurs.

**Supprimer le zoom sur l'image :** le zoom est utile pour diminuer l'effet de l'ouverture numérique avant la LPE. Il faut trouver l'opérateur équivalent sur l'image non zoomée. Une hypothèse à creuser est que ce serait équivalent à une ouverture avec un élément structurant en croix (?).

**Auto-contrôle de la détection des fibres :** savoir si la détection est correcte, ou si au cours des acquisition on arrive à un moment où il faut la refaire.

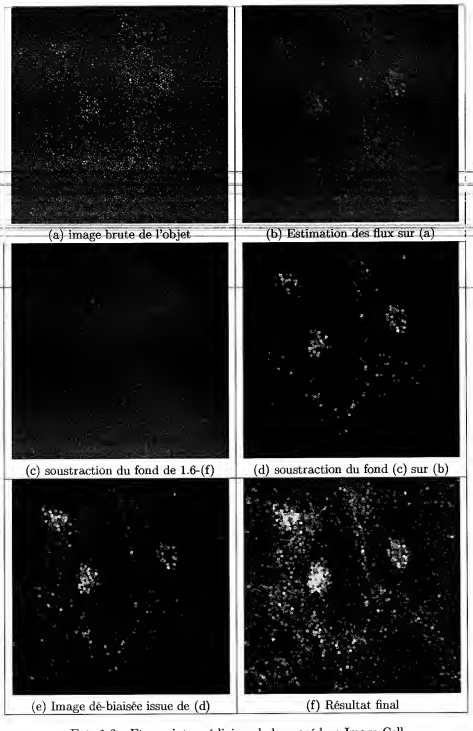


(a) Image initiale des fibres (b) composantes connexe des fibres

(c) Estimation des flux sur (b) (d) Soustraction du fond estimé sur (c)

(e) Image dé-biaisée issue de (d) (f) fond de l'objet (suite sur fig. 1.6)

FIG. 1.5 – Etapes intermédiaires de la procédure Image-Cell



(a) image brute de l'objet (b) Estimation des flux sur (a)

(c) soustraction du fond de 1.6-(f) (d) soustraction du fond (c) sur (b)

(e) Image dé-biaisée issue de (d) (f) Résultat final

FIG. 1.6 – Etapes intermédiaires de la procédure Image-Cell

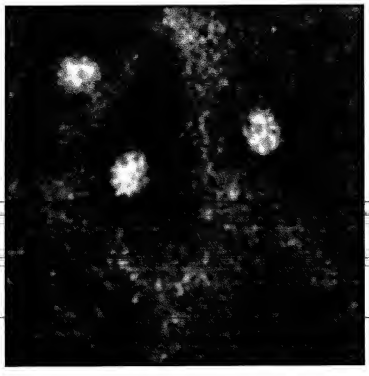


FIG. 1.7 – Résultat final après reconstruction (cetteimage a juste été lissée, et n'est pas une reconstruction par RBF).

**Auto-contrôle de valeur du fond :** savoir sir le fond proposé est valable ou si il faut l'estimer. Dans le cas où le fond n'est pas fournit, en faut-il un ?

**Amélioration continue de la détection des fibres :** on devrait pouvoir utiliser les images des objets pour renforcer ou corriger la détection des fibres.

**Estimer le biais avec des splines :** le biais est estimé par rééchantillonnage bilinéaire qui produit un effet de blocs qui peut être gênant

On peut aussi prévoir plusieurs axes de recherche pour des fonctionnalités futures : ce sera le sujet d'un prochain mémo...

# Chapitre 2

# Programmation d'Image-Cell

## Introduction

Cette partie rassemble les principaux éléments de la librairie. La description de ces élément est globale, cette documentation venant en complément de la documentation automatique[1] décrivant toutes les classes et toutes les fonctions, et des tests unitaires décrivant l'utilisation des classes et fonctions principales.

## 2.1 Une librairie sur les graphes

La programmation des algorithmes a nécessité l'utilisation de structure de graphes. Nous avons donc recherché des librairies C++ permettant de les manipuler. Nous avons testé les librairies suivantes.

**GTL :** pour Graph Template Library (`http://www.infosun.fmi.uni-passau.de/GTL/`). Cette librairie est la plus adaptée à nos besoins et a été celle utilisée pour le prototypage dans [?]. Le problème principal de cette librairie est son prix dissuasif pour les entreprises : 5 000 Euros (licence de développement pour un seul poste). Elle était gratuite pour le stage car utilisée par une étudiante... Nous n'avons donc pas choisi de garder cette librairie pour cette raison.

**LEDA :** pour Library of Efficient Data Types and Algorithms (`http://www.mpi-sb.mpg.de/LEDA/`). Cette librairie traite de nombreux problème de type de données, dont celui des graphes. Cette librairie a cependant beaucoup évoluée vers une sorte de librairie couteau suisse qui fait beaucoup de choses, dont les interfaces graphiques, et qui redéfinit de nombreux outils maintenant standard, notamment les types STL. Il existe donc de nombreux cas de conflits potentiels avec nos softs. Le prix de cette librairie est de 1500 US$ pour un développeur, et 8600$ pour développer sur tout un site, et vendre des produits derrière. J'ai personnellement utilisé cette librairie pour ma thèse (gratuitement car pour une thèse...).

**BOOST :** une sorte de groupware qui travail sur des proposition de futurs standard C++ (`http://www.boost.org/index.htm`). Il y a en fait toute un librairie avec de nombreux thèmes. Le point central est que c'est de la programmation avancée, générique, réutilisable, versatile. C'est beau mais c'est très compliqué à utiliser malgré une documentation très propre. Le problème vient des pré-requis indispensables en programmation objet de haute voltige qui sont un peu trop complexes pour nos besoins, et qui nécessiterai un travail supplémentaire important. Cette librairie est cependant gratuite, et très bien faite. Un attrait intéressant est qu'elle n'est constituée que de fichiers .h, sans partie à compiler. Tout repose

[1] Utilisant doc++

sur une gestion complexe de templates qui ne peuvent pas être précompilée. Il n'y a donc pas de problèmes de compilation sur différentes plateformes.

Toutes ces librairies fonctionnent sous toutes les plateformes courantes, et avec tous les compilateurs courant. De plus elles sont maintenues et stables. Il existe de nombreuses autres petite librairies à droite à gauche mais qui sont loin d'approcher la qualité de celles-ci. Nous n'avons cependant pas pu en retenir une celle, soit à cause du prix et des risques liés à l'utilisation d'une grosse librairies supplémentaire, soit à cause de la complexité d'utilisation trop grande.
Nous avons donc choisi de prendre nos propres structures de données, adaptés aux traitement pour plus de rapidité à la fois de développement et d'exécution. Ces structures de données sont cependant assez souples car elle fonctionnent sur des types et des algorithmes STL qui peuvent être interchangeables assez rapidement...

## 2.2 Gestion générale des paramètres d'algorithmes

### 2.2.1 L'objet paramètre

L'objet paramètre n'existe pas encore en tant que tel, mais va bientôt voir le jour... Il est pour le moment à l'état de concept dans le même sens que la terminologie STL.
Une classe paramètre est de type suivant :

```
class AlgoParam {

  public :

  AdjGraphParam(){
    defaults();
  };

  void defaults() { // Cette fonction fixe les paramètres par défaut

  }

  /// Load parameters from a file. If any error occurs, default values are provided.
  void loadParam(const char * fname);

  /// Save parameters in a file.
  void saveParam(const char * fname);

  /// Output stream
  friend std::ostream & operator << (std::ostream &, AlgoParam &);

  /// Input stream
  friend std::istream & operator >> (std::istream &, AlgoParam &);

public: // parameters list here, with comments please !

[...]

};
```

Les fonctions loadParam et saveParam vont juste créer un flot dans le fichier dont le nom est spécifié. Ils délèguent donc le travaille aux flots `ostream` et `istream`.

Pour le moments les flots sont simples, et ne font que écrire le nom du paramètre suivie de la valeur numérique, le tout sur une ligne. Les paramètres sont encadrés par

```
ALGOPARAM_CONFIGURATION
PARAMETERS
```

et

```
END_ALGOPARAM_CONFIGURATION
```

Ce mécanisme (surement perfectible...) permet de déléguer les entrées et les sorties aux flots qui peuvent donc être imbriqués. Ainsi un objet paramètre peut contenir un autre objet paramètre sans avoir à récrire les entrées et les sorties de cet objet.
De telles classes de paramètres sont utilisés pour `ImageCell` `FiberDetector` `AdjGraph` et `FiberFlow`.

### 2.2.2 Le futur des paramètres

La première chose à faire est de créer une classe virtuelle (c'est-à-dire une interface) pour la classe de paramètres.
Enfin, dans un second temps est à l'étude l'utilisation d'XML pour interfacer ces paramètres.

## 2.3 La détection des fibres : FiberDetector et AdjGraph

Voir le test unitaire `test_fiberdetect` pour les détails d'utilisation.
L'objet FiberDetector est un objet-algorithme modulaire pour détecter les fibres. Il contient un objet spécialisé AdjGraph qui permet de faire l'étape de fusion des fibres qui est la plus compliquée.

### 2.3.1 Fonctionnement détaillé

On retrouve les étapes suivantes dans la classe :

```
void init(Inrimage *inrIn);
void preprocessing();
void watershed();
int splitBiggest ();
int mergeSmallest ();
```

L'étape d'initialisation `init` permet de fixer l'image servant de base à la détection des fibres. Puis l'étape `preprocessing` fait es premiers traitements : zoom, ouverture numérique, inversion de l'image, et quelques calculs préliminaire, comme une structure de voisinage des fibres. Ces deux étapes permettent de préparer l'image pour les traitements de segmentation.
La segmentation est réalisée par une ligne de partage des eaux (LPE) dans la fonction `watershed`. L'algorithme pourrait également interfacer un autre algorithme...
Les deux étapes suivantes sont deux étapes de post-processing de *split and merge*. `splitBiggest` va repérer les fibres trop grosses, puis les resegmenter. `mergeSmallest` permet de fusionner les fibres les plus petites. Cette dernière étape plus complexe et plus critique est elle-même encapsulée dans une classe algorithme AdjGraph expliquée dans la prochaine partie.
Ces deux étapes contiennent une dizaine de paramètres qui permettent d'ajuster la précision et la sensibilité de l'algorithme. La structure de l'algorithme permet d'éventuellement boucler ces étapes afin de faire converger l'algorithme vers une segmentation stable au sens de l'idempotence du *split and merge*, pour un jeu de paramètre donné[2].

[2]Nous n'avons pas de preuve de l'existence d'une telle convergence et il semble difficile *a priori* d'en obtenir.

### 2.3.2 L'étape de fusion avec AdjGraph

Cette partie décrit l'étape de fusion. Cette étape est importante car les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final. La complexité de cette partie nos a obligé à en faire une partie modulaire, au même titre que le `FiberDetector`. Les différentes «modules» ont été les suivants :
- type de calcul de la compacité d'une région, et de la fusion de deux régions
- construction du graphe d'adjacence dédié
- structures de données dédiée aux calculs des attributs utiles.

Cette étape était la plus longue sur le prototype de [?], et a donc fait l'objet d'optimisation. L'étape de fusion n'est réalisé que sur des critères de formes, sans lien directe nécessaire avec l'image de départ[3]. Le problème principal est le calcul des attributs utilisés : la compacité et la surface. La principale difficulté est liée à ce que le graphe va évoluer par fusion de régions, et que l'on cherche à mettre à jour ces attributs ; on veut également tester les fusions possibles et donc pré-calculer les attributs possibles.
Pour cela, on distingue les frontières extérieures et intérieures d'un objet comme indiqué sur la figure 2.1. On divise la frontière d'un objet en fonction de l'adjacence. La frontière de A est égale à la somme des frontière intérieures de A avec tous ses voisins. La frontière intérieure de A commune avec F est égale à la frontière extérieure de F commune avec A. Cette dernière propriété permet de ne parler que de frontière intérieure.

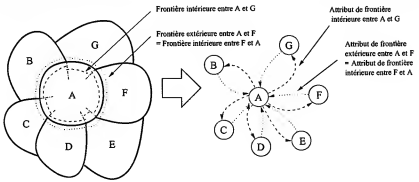



FIG. 2.1 – Les frontières extérieurs et intérieures d'un objet. La frontière intérieure de A commune avec F est égale à la frontière extérieure de F commune avec A. Ces informations sont portés par des arcs du graphe d'adjacence.

Ces informations sont portés par des arcs du graphe d'adjacence. Un arc allant de A vers B va porter l'information de frontière commue que A détient avec B.
On peut formaliser ces relations ainsi. Soit $G = (N, E)$ un graphe avec $N$ l'ensemble de ses noeuds et $E \subseteq N \times N$ l'ensemble de ses arcs. G est un digraphe[4] où deux nœuds n'ont soit pas d'arcs en commun, soit deux arcs antisymétriques. $E$ est donc un ensemble de paires ordonnées. Soit $s : N \to \mathbb{R}^+$ une fonction permettant d'obtenir la surface. Soit $f : E \to \mathbb{R}^+$ une fonction permettant d'obtenir la frontière intérieure entre les deux nœ désigné par l'arc. Enfin soit $c : N \to \mathbb{R}^+$ la fonction permettant d'obtenir la compacité d'une région. On note $\mathcal{V}(n)$, $n \in N$ l'ensemble des voisins de $n$. La compacité peut alors s'écrire :

$$c(n) = \frac{\left(\sum_{m \in \mathcal{V}(n)} f((n,m))\right)^2}{s(n)}. \tag{2.1}$$

[3]Ce qui est en fait une extension (lourde !) possible
[4]un graphe où tous les arcs sont dirigés

Pour tester la compacité de la fusion éventuelle de $n$ et $m$, on note la nouvelle compacité $c(n \cup m)$ :

$$c(n \cup m) = \frac{\left(\sum_{o \in \mathcal{V}(n)} f((n,o)) + \sum_{o \in \mathcal{V}(m)} f((m,o)) - f((n,m)) - f((m,n))\right)^2}{s(n) + s(m)}. \quad (2.2)$$

Et si la fusion de $m$ avec $n$ est effectuée, on modifie les fonctions (on suppose que $m$ est rajoutée à $n$ ) :

$$s(n) \leftarrow s(n) + s(m) \quad (2.3)$$

$$f(n,o) \leftarrow f(n,o) \text{ pour } o \in \mathcal{V}(n) \text{ et } o \notin \mathcal{V}(m) \quad (2.4)$$

$$f(n,o) \leftarrow f(n,o) + f(m,o) \text{ pour } o \in \mathcal{V}(n) \cap \mathcal{V}(m). \quad (2.5)$$

On peut ainsi ne travailler que sur les graphes, sans avoir à revenir à la segmentation initiale pour recalculer les attributs lors d'une fusion. Cela nous a permis de tester beaucoup plus de fusion, voire éventuellement de tester toutes les fusions possibles.

## 2.4 Structures de données associées aux fibres : FiberStructure et FiberInfo

Voir le test unitaire `test_fiberInfo` pour les détails d'utilisation.

### 2.4.1 Lien entre la structure des fibres et les informations sur chaque fibre

La structure de donnée est spécifique au problème qui nous intéresse. Les informations qui nous intéressent sont des informations attachées aux fibres seulement, les fibres étant en fait liées à la structure détectée. Les deux types d'informations sont donc étroitement liées, les informations sur les fibres ne pouvant exister indépendamment de la structure des fibres.

#### Modèle sujet-observateur

Nous avons donc choisi de les relier sous le modèle de l'observateur des design patterns [?]. Nous avons donc défini ce mécanisme dans le fichier `Observer.h`. Le principe est le suivant :

> «Définit une interdépendance un à plusieurs, de façon telle que, quand un objet change d'état, tous ceux qui en dépendent en soient notifiés et automatiquement mis à jour.»

Ainsi on distingue le sujet, qui pour nous est la structure des fibres détenue par la classe `FiberStructure` ; et les observateurs qui sont les informations que l'on peut attacher aux fibres (classe `FiberInfo<T>`).
On a ajouté un comportement supplémentaire : un objet `FiberInfo<T>` ne peut exister sans un `FiberStructure`, ce pourquoi il n'existe qu'un constructeur qui prend en argument la `FiberStructure`. L'objet `FiberInfo<T>` va d'ailleurs s'initialiser tout seul en regardant la structure à laquelle il est attaché.
Quand la structure vient à changer, tous les `FiberInfo<T>` en sont informés si besoin, et ceux-ci vont alors se réinitialiser automatiquement. Quand la structure est détruite, les objets d'information sont informés également qu'ils ne sont plus valides.

#### Structure codée comme un graphe : FiberStructure

La classe fiberstructure se construit à partir de l'image des composantes connexes. L'objet va alors garder en interne une copie de cette image de référence. Elle contient en interne la structure de graphe associée alors aux composantes connexes. Elle contient également d'autres informations, comme les barycentres des régions, en coordonnées pixel, les surface des régions ...

Toutes ces informations sont accessibles depuis l'interface de la classe, mais ne sont pas modifiables, car elles dépendent de la structure même.

#### La classe d'information : FiberInfo

La classe `FiberInfo<T>` est une classe template car les informations peuvent être de nature très différentes. Elle contient un vecteur STL d'information dont la taille est gérée automatiquement en fonction de la structure associée. Ce vecteur est un membre ***public*** afin d'avoir des accès rapides aux informations. La classe sert juste de décorateur[5] pour pouvoir gérer les dépendances avec la structure des fibres.
Quand le type du template est numérique, il existe des fonctions pour additionner, soustraire, diviser et multiplier deux `FiberInfo<T>`, ou bien un `FiberInfo<T>` avec un élément de `T` : `addFI`, `subFI`, `divFI`, `mulFI`.

### 2.4.2 Transformation d'images en FiberInfo et réciproquement

Voir le test unitaire `test_fiberProcessing` pour les détails d'utilisation.
Une image peut être transformée en `FiberInfo<float>` avec la fonction d'estimation simple des flux :
`int flowEstimation (Inrimage* image, FiberInfo<float> & flow, bool bias = false)`.
Il faut au préalable initialiser de `FiberInfo<float> flow` avec la structure, ainsi l'algorithme connaît via `flow` l'image des composantes connexes à utiliser.
La transformation réciproque est la reconstruction d'image, pour laquelle il n'existe pour le moment que la méthode de mosaïquage :
`int imageMosaic (Inrimage* image, FiberInfo<float> & flow)`
Le principe est similaire à `flowEstimation`. La structure est passée via le `flow`.

## 2.5 Soustraction du fond et biais : FiberFlow

Voir le test unitaire `test_fiberFlow` pour les détails d'utilisation.

### 2.5.1 Fonctionnement général

La classe FiberFlow implémente le bloc d'estimation des flux de la partie 1.1, sauf la partie de transformation de l'image en `FiberInfo`. La classe comporte deux méthodes principales : `setBackground` et `setImage`, et son comportement est paramétré avec la classe de paramètre `FiberFlowParam`. La classe permet de soustraire le fond qui est donnée, ou bien il peut l'estimer. La classe sert aussi à calculer le biais dans l'image, après soustraction du fond.

### 2.5.2 Le calcul du fond

Voir le test unitaire `test_fiberHisto` pour les détails d'utilisation.
Pour calculer le fond, on calcul un quantile de l'histogramme des fibres. Pour cela il existe une classe `FiberHisto<T>` qui prend un `FiberInfo<T>` en constructeur, et qui permet de faire des calculs sur l'histogramme calculé à partir de l'information des fibres. Ces calculs sont plus rapides que sur l'image elle-même.

[5] Attention, pas au sens du Design Pattern décorateur [?] !

### 2.5.3 Le calcul du biais

Voir le test unitaire `test_fiberProcessing` pour les détails d'utilisation.
Le calcul du biais n'est en fait pas calculé sur l'image, mais encore une fois sur l'information des fibres directement. C'est encore un cas où on utilise la structure et l'information des fibres. Ce calcul est fait dans une fonction :

```
int biasEstimate  ( FiberInfo<float> & flowIn,
                    FiberInfo<float> & bias,
                    int nbBlocks,
                    biasEstimatorType bet=BET_MEAN)
```

L'estimateur du biais est pour le moment à choisir entre la moyenne (`BET_MEAN`) et la médiane (`BET_MEDIAN`).
On divise l'espace total des fibres en `nbBlocks` blocs de taille égale. On parcourt ensuite l'ensemble des blocs et on trie les fibres en fonctions du bloc dans lequel elles tombent. Puis, pour chaque bloc, on applique l'opérateur choisi d'estimation du biais.
On calcule le biais ensuite en interpolant pour chaque fibre uniquement (et non pour toute l'image) les valeurs des blocs les plus proches. On utilise pour le moment une interpolation bilinéaire, car c'est la seule disponible avec Inrimage. On fait une proposition pour avoir une interpolation spline cubique.

## 2.6 Améliorations

- La classe de paramètres est à optimiser et à programmer plus proprement.
- Le biais devrait être interpolé avec une spline cubique
- Une librairie de graphe serait tout de même la bienvenue. Cela permettrait d'avoir une structure commune entre `FiberDetector`, `AdjGraph`, et `FiberStructure`, ce qui n'est actuellement pas exactement le cas.

# Chapitre 3

# Utilisation d'Image-Cell

C'est une partie délicate car elle doit contenir le savoir-faire acquis pendant le prototypage, le développement et les nombreux tests.
Les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final.

## 3.1 Utilisation de la paramétrisation

Il faut ouvrir une fenêtre d'un éditeur de texte (notepad, emacs, wordpad...) avec le fichier imagecell.txt. Ce fichier est relu à chaque utilisation. La liste des paramètre, ainsi que des résumés d'indications sont données en annexe A.

## 3.2 Paramétrage de FiberDetector

### 3.2.1 Prétraitements

Dans presque tous les cas, il faut faire le zoom sur l'image pour que l'algorithme fonctionne correctement.

### 3.2.2 Paramétrage de AdjGraph

Les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final.

## 3.3 Paramétrage du FiberFlow pour la calibration

## 3.4 Paramétrage du FiberFlow pour l'image à reconstruire

## 3.5 Paramétrage d'ImageCell : les reste des paramètres

# Annexe A

# Paramètres de Image-Cell

Les paramètres internes d'Image-Cell sont modifiables via un fichier de paramètres présent dans le répertoire source du lancement de l'application, au même endroit que tomoscope.cfg. Le fichier des paramètres `imagecell.cfg` a la forme suivante :

| Ligne du fichier | Défaut | Possibles | Remarques |
|---|---|---|---|
| IMAGECELLPARAM_CONFIGURATION | | | Paramètres de l'algorithme entier |
| PARAMETERS | | | - |
| FIBERDETECTORPARAM_CONFIGURATION | | | Paramètres de l'algorithme entier |
| PARAMETERS | | | - |
| _doZoom_ | 1 | $\{0,1\}$ | Travail sur l'image doublée (plus précis) |
| _doDiffusion_ | 0 | $\{0,1\}$ | Option non pris en compte pour le moment. A régler. |
| _hDomeWatershed_ | 1 | $\{1,..,10\}$ | Hauteur des maxima sélectionnés pour initialiser les fibres. |
| _oversizedFactor_ | 1.8 | $[1;2]$ | Taille normalisée (par rapport à la moyenne) au delà de laquelle une fibre va être divisée. Ce critère est utilisé avec le critère suivant sur la taille (ET logique) |
| _overSizedNeighLimit_ | 6 | $\{5,6,7\}$ | Nombre de voisins au delà duquel une fibre va être divisée. Ce critère est utilisé avec le critère précédent sur la taille (ET logique) |
| _overSizedResegmentMethod_ | 1 | $\{0,1\}$ | 0 : resegmenter dans une fonction distance ; 1 : resegmenter dans l'image originale sans prétraitement |
| ADJGRAPHPARAM_CONFIGURATION | | | Paramètres pour la fusion des fibres |
| PARAMETERS | | | - |
| _underSizedFactor_ | 0.6 | $[0.1;1]$ | Taille normalisée (par rapport à la moyenne) en deçà de laquelle une fibre va être une candidate pour la fusion. Ce critère est utilisé avec le critère suivant sur la taille (ET logique) |
| _underSizedMendatoryMergeSlope_ | $-0.22$ | $[?;?]$ | Ce paramètre et le suivant déterminent une droite dans l'espace (nombre de voisins)x(taille normalisée) délimitant les fibres qui doivent obligatoirement être fusionnées. *Mieux vaut ne pas y toucher sans faire de calcul précis !* La droite par défaut passe par les points $(6;1/3)$ et $(3,1)$. Augmenter la pente revient à s'occuper plus des grosses fibres avec peu de voisins. |
| _underSizedMendatoryMergeOffset_ | 1.667 | $[?;?]$ | Voir remarque précédente. Ce paramètre est l'ordonnée à l'origine (axe des ordonnées=tailles normalisées). L'augmenter revient à obligatoirement fusionner plus de fibres. |
| _underSizedFilterSize_ | 1 | $\{0,1\}$ | Doit-on filter les fusions possibles en fonction de la taille totale de la fusion ? |
| _underSizedFilterSizeMax_ | 1.17741 | $[1;3]$ | Si oui, interdire les fusions donnant une taille de fibre normalisée supérieure à |
| _underSizedFilterCompact_ | 0 | $\{0,1\}$ | Filtrage des fibres fusionnées en fonction de la compacité |
| _underSizedFilterCompactMax_ | 2 | $[1;10]$ | Facteur multiplicatif de l'écart-type |

| Ligne du fichier | Défaut | Possibles | Remarques |
|---|---|---|---|
| _underSizedFilterMoreCompact_ | 0 | {0, 1} | Ne garde que les fusions qui rendent les fibres fusionnées plus compacte que la fibre voisine seule. |
| END_ADJGRAPHPARAM_CONFIGURATION | | | |
| END_FIBERDETECTORPARAM_CONFIGURATION | | | |
| FIBERFLOWPARAM_CONFIGURATION | | | Paramètres d'estimation du flux pour l'image de calibration |
| PARAMETERS | | | - |
| _backgroundMode_ | 1 | {0, 1, 2} | 0 : pas de soustraction du fond ; 1 : soustraction avec le fond fourni, sinon 2 ; 2 : estimation du fond sur l'histogramme |
| _backgroundEstimationCut_ | 0.01 | [0; 0.1] | Quantile de l'histogramme de l'image pour l'estimation du fond |
| _biasCorrection_ | 1 | {0, 1} | Correction du biais, après soustarction du fond |
| _biasCorrectionNbBlocks_ | 16 | {2, .., 64} | Si correction du biais, nombre de blocks en X et en Y pour son estimation |
| _biasCorrectionOperator_ | 1 | {0, 1} | Opérateur pour la correction du biais : 0 : moyenne ; 1 : médiane |
| END_FIBERFLOWPARAM_CONFIGURATION | | | |
| FIBERFLOWPARAM_CONFIGURATION | | | Paramètres d'estimation du flux pour l'image à reconstruire |
| PARAMETERS | | | - |
| _backgroundMode_ | 1 | {0, 1, 2} | 0 : pas de soustraction du fond ; 1 : soustraction avec le fond fourni, sinon 2 ; 2 : estimation du fond sur l'histogramme |
| _backgroundEstimationCut_ | 0.01 | [0; 0.1] | Quantile de l'histogramme de l'image pour l'estimation du fond |
| _biasCorrection_ | 1 | {0, 1} | Correction du biais, après soustarction du fond |
| _biasCorrectionNbBlocks_ | 4 | {2, .., 64} | Si correction du biais, nombre de blocks en X et en Y pour son estimation |
| _biasCorrectionOperator_ | 1 | {0, 1} | Opérateur pour la correction du biais : 0 : moyenne, 1 : médiane |
| END_FIBERFLOWPARAM_CONFIGURATION | | | |
| _tempoFilterTypeFib_ | 3 | {0, .., 4} | Type du filtre temporel pour l'estimation des fibres :0 : moyenne ;1 : moyenne coupée ; 2 : médianne ; 3 : max ; 4 : min |
| _calibration_ | 1 | {0, 1} | Effectuer la calibration |
| _outputSigmaFilter_ | 3 | $[0; 10] \cup \{-1\}$ | Ecart type de la gaussiène pour un filtrage passe bas en sortie de image cell. Si la valeur est négative (-1), il n'y a pas de filtrage. |
| END_IMAGECELLPARAM_CONFIGURATION | | | |

# Bibliographie

**Diligence Document 7**

# Manuel de référence du module ImageCell

Auteur : Aymeric Perchant

Date : 2002-04-24

**Diffusion :** interne

**Section :** informatique, image

**Sujet :** Manuel de référence de la partie libMKTProcessing pour imageCell. Ce manuel contient les informations de traitement d'images, de programmation C++, et d'utilisation.

**Version :** *Revision* : 1.3 Attention, cette version est la version du fichier principal. Chaque chapitre possède sa propre version car les chapitres sont indépendants les uns des autres.

**Référence du document :** `$Id: ReferenceImageCell.tex,v 1.3 2002-04-24 07:14:48 aymeric Exp $`

## Introduction

Ce document rassemble les différentes informations de référence pour le module principal intitulé Image-Cell. Ce module contient les parties suivantes de traitement d'images :

- détection des fibres,
- estimation des flux revenant des fibres,
- corrections des défauts de l'appareil (biais, fond, ...),
- calibration de l'appareil,
- reconstruction d'images.

Ces différentes parties seront abordées en détail dans chaque partie pour expliquer les algorithmes et les paramètres (chapitre 1), l'implantation de ceux-ci (2) et leur utilisation pratique (3).

### Corrections

- rajout d'un filtrage passe bas, mars 2002.

# Table des matières

# Chapitre 1

# Traitement d'images dans Image-Cell

## Introduction

**Auteur CVS (du chapitre) :** *Author* : *aymeric*

**Version (du chapitre) :** *Revision* : 1.4

**Référence du document :** `$Id: chapTdi.tex,v 1.4 2002-04-24 07:14:48 aymeric Exp $`

Cette partie décrit les algorithmes en commençant par la description générale, jusqu'aux boites élémentaires.

## 1.1 Schéma global

La figure 1.1 représente le schéma global du traitement. On peut distinguer quatre blocs, dont deux sont identiques, à une paramétrisation près. Il existe donc trois groupes de traitements qui sont les suivants :

**Détection des fibres :** ces traitements permettent de détecter et d'isoler chaque fibre sur une image, ainsi que d'analyse la structure d'agencement des fibres du guide d'images ;

**Estimation des flux :** une image brute de taille $640 \times 640 \times 20$ (largeur, hauteur, nombre d'images temporellement )contient 8 méga pixels qui représentent l'information vue par 10000 ou 30000 fibres. Ce bloc permet d'isoler l'information effectivement vue par chaque fibre ;

**Calibration et reconstruction :** La connaissance de l'information vue par chaque fibre est ensuite traitée pour être reconstruite sous la forme d'une image débarrassée des défauts de l'appareil.

Chacun de ces trois blocs sont maintenant détaillés. Le prototypage des algorithmes décrits ici est détaillé dans le rapport de stage de Sandra Marti [?] ; nous renvoyons le lecteur à cette référence pour plus de renseignements sur la démarche de développement des algorithmes, notamment celui de détection des fibres.

Le bloc de gauche d'estimation des flux permet de mesurer le flux sur une image qui représente un objet aux propriétés constantes dans l'espace (un milieu diffusant homogène ou un objet presque homogène en mouvement aléatoire, un miroir). La sortie de ce bloc est donc une image des taux d'injection dans chaque fibre, et qui en prend en compte l'intégralité de la chaîne.

Le bloc de droite d'estimation des flux permet de l'estimer pour un objet à observer, et à travers l'appareil. Le dernier bloc de calibration est une division de l'image des taux d'injection par l'image de l'objet observé. Cette opération permet de compenser les mauvaises injections dans certaines fibres. Puis l'image est reconstruite.

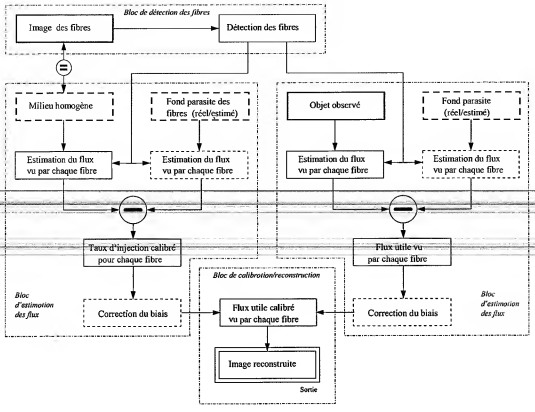



FIG. 1.1 – Schéma global du traitement. Les cadres gras représentent les entrées. Les cadres pointillés représentent les données ou les traitements facultatifs (biais, soustraction du fond...).

## 1.2 Bloc de détection des fibres

La détection des fibres s'organise autour de quatre traitements successifs :
- prétraitements,
- ligne de partage des eaux (LPE) : la première segmentation par région,
- séparation des fibres trop grosses (split),
- fusion des fibres trop petites (merge).

Les deux dernières étapes peuvent être bouclées ; néanmoins une analyse de performance et de convergence doit alors être menée. Nous n'aborderons pas ce sujet pour le moment.

### 1.2.1 Prétraitements

Les prétraitements réalisés sont les suivants.

**Diffusion anisotrope :** le but est de lisser l'image dans les zones plates, c'est à dire les zones inter fibres. Le prototype fonctionnait mais n'utilisait pas la librairie inrimage, mais une autre libriairie (celle utilisée par Sandra, issue de l'ENST). Le réglage de la diffusion est laissée à plus tard, lors de la prochaine release de la libInrimage qui contiendra de la documentation à ce sujet ! Paramètre `[_doDiffusion_]`.

**Interpolation $\times 2$ au plus proche voisin :** on cherche à simuler des éléments structurant de morphologie mathématique avec un rayon inférieur à un. L'image est doublée pour que l'ou-

verture qui suit ne touche pas aux maxima isolés, mais seulement ceux qui sont 8-connexe, mais non 4 connexe (voisins par une diagonale). L'intérêt est de faire une sélection des maxima éliminé par l'ouverture. Paramètre [_doZoom_].

**Ouverture numérique :** on cherche à éliminer les maxima parasites situés sur les fibres. C'est un prétraitement classique de la LPE, mais éventuellement modifié par le comportement du doublage de l'image expliqué précédemment.

**Inversion de l'image :** la LPE fonctionne à partir de minima, on inverse donc l'image pour transformer les minima en maxima.

L'opération d'interpolation et d'ouverture qui suit devrait pouvoir être remplacé par un nouvel opérateur de morphologie mathématique qu'il faudrait définir. Cela permettrait un gain important, puisque toutes les opérations suivantes se feraient sur une image de taille inférieure.

### 1.2.2 Premier watershed

Le watershed de la libInrimage nécessite la définition de marqueurs dans l'images. Nous obtenons les marqueurs comme minima de l'image prétraitée avec de h-dômes [7]. Les h-domes pour les minima sont définis pour un entier $h$, une fonction $f : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ représentant l'image concernée :

$$g = E_f^\infty(f + h) - f, \tag{1.1}$$

avec $E_a^\infty(b)$ la reconstruction géodésique par érosion de $b$ dans $a$. $g$ désigne ici l'ensemble des h-domes, que l'on seuil à 1 afin de tous les récupérer. On obtient ainsi les minima de profondeur au moins égale à $h$. Ce paramètre $h$ est donné par [_hDomeWatershed_].
Le résultat de cette LPE est une image de composantes connexes représentant chaque fibre détectée. Afin de résoudre bon nombre de problèmes de bord on colle les composantes touchant le bord au bord lui-même. Ces composantes sont considérées comme faisant partie du bord, et du fond et sont mise à la composante connexe d'indice 0, qui est l'indice réservé au fond, et au bord.
Enfin, les composantes connexes sont réorganisées pour ne pas présenter de trous, et sont triés par ordre décroissant de taille.

### 1.2.3 Split

Les prétraitements, et notamment l'ouverture numérique servent à limiter le défaut principal de la LPE qui est de sursegmenter. En faisant cela, on a tendance à augmenter le nombre de segments qui vont englober plusieurs objets (ici, des fibres). Le premier défaut que l'on corrige est la sous-segmentation inévitable au traitement, mais surtout rajoutée par ces prétraitements. Pour cela on effectue les traitements suivants.

**Sélection des segments suspectés de sous-segmentation :** on trouve les fibres dont la taille est supérieure à un seuil fixé en fonction de la moyenne des tailles normalisées (paramètre [_oversizedFactor_]), et en fonction du nombre de voisins (paramètre [_overSizedNeighLimit_]).

**Re-segmentation de ceux-ci :** Les segments sélectionnés sont isolées et re-segmenter soit sur l'image initiale sans prétraitement, soit sur l'image de carte de distance à l'intérieur de ces régions (paramètre [_overSizedResegmentMethod_]). Dans le premier cas, on suppose que les défauts majoritaires proviennent des prétraitement, dans le second cas on suppose qu'ils étaient là avant, et que l'on souhaite séparer les segment aux endroits de rétrécissement (forme de la cacahuète). La LPE est effectuée avec les même paramètres que la première.

### 1.2.4 Merge

Puis on cherche à corriger le défaut le plus classique (pour une LPE) : la sursegmentation. On cherche donc à fusionner les segments. Pour cela on effectue les opérations suivantes :
– présélection des *candidats à la fusion*,

- parmi ceux-ci trier ceux qui seront obligatoirement fusionnés,
- pour les autres, éliminer les fusions impossibles
  - répertorier les fusions possibles
  - éliminer les fusions donnant de mauvais résultats
  - si il ne reste pas de fusion après filtrage, retirer la fibres des fibres à fusionner
- pour toutes les fibres restantes, prendre la meilleur fusion au sens de la compacité des segments fusionnés.

Les points importants sont détaillés ci-après.

### Remarques sur la compacité

La compacité utilisée est le critère le plus simple existant. Pour un objet de périmètre $P$ et de surface $S$, le critère est $\frac{P^2}{S}$. Le périmètre est pris comme étant la frontière de l'objet 4-connexe, et donc est défini comme tous les pixels de l'objet ayant un voisin 8-connexe avec le fond (défini ici comme étant tout le reste de l'image. La surface est simplement l'objet lui-même.
Cependant, ce critère simple à calculer n'est pas invariant avec l'échelle de l'objet. On peut montrer que pour des formes simple, le critère est croissant avec la taille de l'objet au lieu d'être invariant. Cette croissante n'est cependant qu'en moyenne en $\frac{1}{n^2}$, avec n la diamètre de l'objet (ici, pour une boule en 4-connexité). Ce critère reste donc tout de même assez raisonnable, mais plus délicat à manier dans le cas de la comparaison de la compacité d'objets de taille différentes.
Il faut remarquer ici que les premiers essais sont réalisés sur une image déformée par les distortions, et qu'il faudrait, pour être plus précis soit corriger l'image avant, soit tenir compte de l'anisotropie non stationnaire spatialement : on a tendance à surestimer à la fois les frontières et la surface sur les bords de l'image.

### Sélection des fibres détectées trop petites

La sélection s'effectue en deux étapes. On commence par trouver les segments candidats à la fusion avec un seuil sur la taille. On utilise le paramètre `[_underSizedFactor_]` qui est un facteur multiplicatif de la moyenne, et est donc inférieur à 1 en général. La seconde étape est de trouver parmi ces segments lesquels doivent obligatoirement être fusionnés, et lesquels doivent être filtrés.

### Les segments obligatoirement fusionnés

La sélection des segments (ou fibres) obligatoirement fusionnés s'effectue dans l'espace (nombre de voisins) $\times$ (taille normalisée). La taille est normalisée par rapport à la moyenne (on divise toutes les tailles par la moyenne des tailles). Le nombre de voisins est compté avec la 8-connexité.
Les figures 1.2 et 1.3 représentent l'histogramme conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. On remarque qu'il existe un axe principal qui est centré sur la droite passant par des points d'intérêts particuliers $(6,1)$, $(8,2)$, et $(5,.5)$. Ces points peuvent s'explique géométriquement, comme indiqué sur la figure 1.4
Cet axe est donc un axe naturel de la structure hexagonale des fibres dans cet espace. En s'orientant sur cet axe, les petites fibres se situent vers un nombre de voisins faibles et une taille faible. On détermine une frontière linéaire pour sélectionner les fibres détectées réellement trop petites. Après quelques tests sur des images réelles de la base de données «Novembre 2001» et en fonction des résultats préliminaires de [?], on a choisi une droite passant par les points $(6;\frac{1}{3})$ $(3;1)$. Les paramètres de cette droites sont données sous la forme $y = mx + p$, avec $m$ la pente `[_underSizedMendatoryMergeSlope_]` et $p$ l'ordonnée à l'origine `[_underSizedMendatoryMergeOffset_]`.
Tout point en dessous de la droite sera fusionné obligatoirement, et ne va donc pas passer par les filtres décrits dans la section suivante. Les autres fibres sont par contre filtrée comme suit.

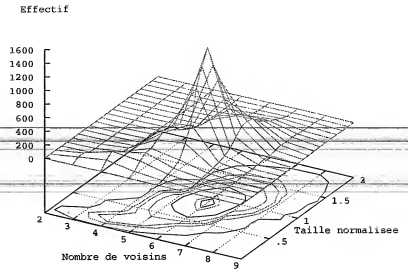



FIG. 1.2 – Vue 3d de l'histogramme 2d conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. Des lignes de niveaux sont projetées sur la fond du graphique : les lignes sont centrées autour de 6 voisins et de la moyenne de la taille des fibres.

**Filtrage des fusions possibles**

On entend par filtrage ici un procédé qui retire certaines fusions. Il existe trois filtres possibles.

**Filtrage sur la taille de la fusion :** (paramètre `[_underSizedFilterSize_]`) on élimine les fusions dont la taille totale dépasse un seuil fixé par le paramètre `[_underSizedFilterSizeMax_]`, qui est pris comme facteur multiplicatif de l'écart type ajouté à la taille moyenne.

**Filtrage sur la compacité totale :** (paramètre `[_underSizedFilterCompact_]`) on élimine les fusions dont la compacité totale dépasse un seuil fixé par le paramètre `[_underSizedFilterCompactMax_]`, qui est pris comme facteur multiplicatif de l'écart type ajouté à la taille moyenne.

**Filtrage sur le changement de compacité :** (paramètre `[_underSizedFilterMoreCompact_]`) on élimine les fusions qui augmentent la compacité de l'objet avec lequel on veut fusionner.

Dans les paramètres par défaut, on n'utilise pour le moment que le premier filtre qui semble suffisant dans la plupart des cas. Les autres filtres semblent trop restrictifs et doivent être encore étudiés.

## 1.3 Bloc d'estimation des flux

Le bloc d'estimation des flux comprend plusieurs sous-parties :

**estimation du flux vu par chaque fibre :** c'est ce que l'on peu récupérer d'une image à l'aide de la détections es fibres ;

**estimation du fond de l'image :** ici le fond désigne les réflexions parasites ou (*ou* inclusif!) l'offset du à l'électronique et au détecteur ;

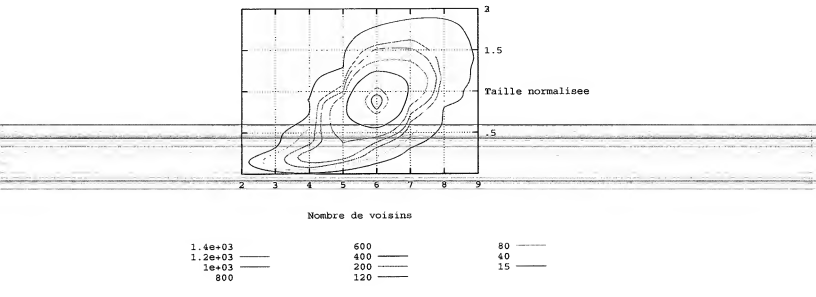



FIG. 1.3 – Lignes de niveau de l'histogramme 2d conjoint de la taille normalisée et du nombre de voisins pour chaque fibre. La forme 3d de cet histogramme est donnée sur la figure 1.2

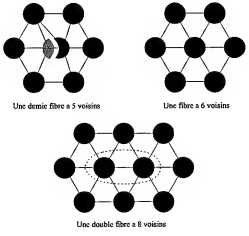



FIG. 1.4 – Variation du nombre de voisins en fonction de la taille du segment détecté comme étant une fibre

**soustraction du fond :** on retire le fond à l'image ;

**correction du biais :** les images sont en général biaisés, c'est à dire qu'il existe un fond lentement variable au lieu d'être constant.

Le bloc de soustraction du fond est optionnel, ainsi que celui de la correction du biais.

### 1.3.1 Estimation du flux vu par chaque fibre

Le rapport [?] décrit une méthode pour *rattraper* les problèmes de saturation de l'image. Ces problème vont disparaître avec l'arrivée de l'électronique mi mai 2002. Nous avons donc décidé de ne pas continuer dans cette voie pour le moment. Ici nous avons simplement choisi d'estimer le flux avec la moyenne sur la fibre.
Plusieurs améliorations sont envisageables :
- tenir compte du bruit poissonien dans l'estimateur du max avec la moyenne,
- tenir compte de la disparité de forme de fibres du guide d'image ; on peut constater que les toutes les fibres ne sont pas identiques et que l'estimateur du flux devrait s'adapter à la forme exacte de la fibre,
- vérifier les problèmes possibles dues à la zone inter-fibre, en fonction de la tache focale sur l'entrée du guide,
- et plus généralement récupérer le modèle complet d'injection et de retour du spot décrit dans [?].

Ces améliorations peuvent être étudiées, mais ne sont pas prioritaires pour le moment.

### 1.3.2 Estimation ou utilisation du fond, puis soustraction

Le fond peut avoir plusieurs sources ; celles-ci sont décrite dans les mémos [?, ?, ?]. En résumé, nous appelons fond ici soit les réflexions parasites sur les optiques, et donc y compris sur la sortie du guide d'images, mais également l'offset due à la chaîne de numérisation.
Si l'offset est dominant sur l'image, on ne peut pas obtenir le fond simplement en retirant l'image, car l'offset dépend du contenu, et n'est donc plus le même. Dans ce cas, on utilise un quantile de l'histogramme ($\frac{1}{100}$ ou $\frac{1}{1000}$) pour l'estimer.
Dans le cas contraire, il faut utiliser le fond acquis lorsqu'on retire l'objet à regarder, et le soustraire. Dans tous les cas il faut bien penser à saturer la soustraction pour ne pas être gêné par des outliers négatifs. L'utilisation d'un talon (cf. [?]) n'est pas retenue car le fond diffus n'est pas du tout une composante majoritaire du signal.

### 1.3.3 Correction du biais

Le biais doit *a priori* être corrigé sur l'acquisition des taux d'injection (branche de gauche) et sur l'acquisition de l'objet (branche de droite). Pour le premier cas, cela vient du fait que la calibration se fait sur un miroir plan, et que la courbure de champ va réduire la qualité d'injection au retour sur les bords (qui sont défocalisés). Ce ne serait probablement pas le cas
Sur l'objet, ou dans un milieu diffusant homogène, l'injection reste moins bonne sur les bords, et cela se traduit par un biais très similaire quant à sa forme au premier.
Dans tous les cas le biais a une symétrie quasi circulaire.
L'estimation du biais se fait en divisant l'image en $N \times N$ blocs de taille fixe, puis en estimant le biais sur chaque bloc. Pour cela il faut considérer la nature de l'objet observé. Dans le cas d'un objet homogène, le biais peut être acquis en prenant la valeur moyenne ou médiane sur le bloc. Quand il y a un objet, il faut savoir si cet objet est plus sombre ou plus claire que le biais. Dans notre cas, le biais est multiplicative, et on prend donc plutôt un opérateur de moyenne ou de médiane (par rapport à un max ou min pour un biais additif).
On obtient alors une image de taille $N \times N$ qui est utilisée avec une interpolation pour trouver la valeur du biais vue par chaque fibre. On utilise pour le moment une interpolation bilinéaire, faute de mieux. La librairie inrimage pourrait être complétée pour intégrer une interpolation en spline cubique point à point.

On peut remarquer que le biais peut être estimée plus finement avec d'autres méthodes plus subtiles qu'il faudrait étudier. Notre méthode reste cependant rapide et semble suffisante.
Une fois le biais estimé, il faut diviser l'image par son biais. Il faut penser à vérifier que le biais est bien supérieur à un pour ne pas multiplier en fait ! Dans le cas de l'image de calibration, on va rétablir une dynamique fictive sur 1000 niveaux afin de ne pas avoir de problèmes avec la seconde division pour la calibration du taux d'injection (cf. partie 1.4.1).

## 1.4 Bloc de calibration et reconstruction

### 1.4.1 Calibration des taux d'injection

Le bloc d'estimation du flux de gauche permet d'obtenir une image des taux de calibration fibre à fibre. On peut dont l'utiliser pour rétablir une injection de 100% sur toutes les fibres. Pour cela, on divise l'image de l'objet par l'image du taux d'injection, après avoir pris quelques précautions sur les valeurs de la calibration (supérieures à 1, avec suffisamment de dynamique).

### 1.4.2 Reconstruction mosaïque

La reconstruction mosaïque s'effectue en répartissant sur toute la surface de chaque fibre la valeur estimée du flux prise après l'image de calibration.
Afin de donner un aspect plus lisse, on a rajouté un filtrage récursif passe bas de type Déricheé. Le paramètre `[_outputSigmaFilter_]` permet de régler l'écart-type. Noter que la complexité de l'algorithme est indépendante de l'écart type de la gaussiène.

### 1.4.3 Reconstruction par RBF

A faire...

## 1.5 Résultats et discussion

Les résultats sont visibles dans la base de données d'images...

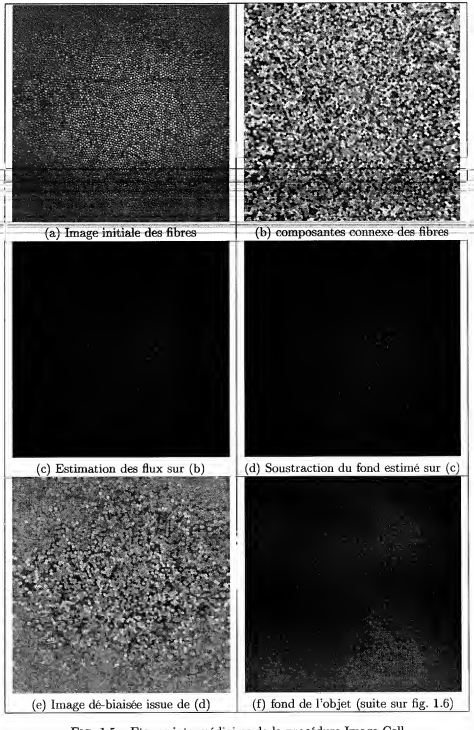
### 1.5.1 Exemple

Les figures 1.5 et 1.6 illustrent toutes les étapes du traitement de l'image. Dans ce cas de figure, le fond de l'image de calibration des fibres est estimé, ce pourquoi il n'apparaît pas sur les images. Le fond de l'objet est par contre fourni. Les images de calibration et l'objet sont toutes les deux dé-biaisées. Ce résultat a été obtenu avec les paramètres par défaut. Sur cet exemple ; on remarque que les images des biais n'ont pas la symétrie circulaire habituelle. La raison est que la majorité du biais venait en fait de la saturation du détecteur qui se traduisait par un nouveau fond additif non prévu au départ, mais qui est tout de même atténué par les traitements.

### 1.5.2 Discussion

De nombreuses idées d'améliorations ont été proposée dans ce chapitre. Certaines semblent plus importantes que d'autres. Voici la proposition d'un tri de ces idées par ordre décroissant d'importance.

**Reconstruction par base de fonctions radiales :** cela permet d'obtenir une image lisse et de faire d'éventuels traitements ultérieurs.

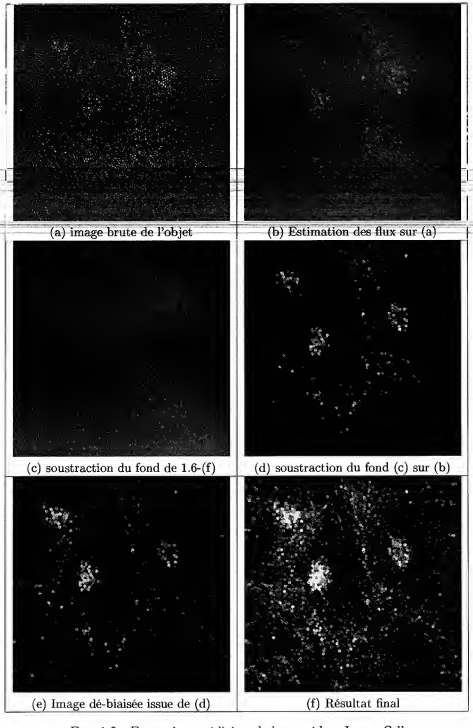


(a) Image initiale des fibres (b) composantes connexe des fibres

(c) Estimation des flux sur (b) (d) Soustraction du fond estimé sur (c)

(e) Image dé-biaisée issue de (d) (f) fond de l'objet (suite sur fig. 1.6)

FIG. 1.5 – Etapes intermédiaires de la procédure Image-Cell



(a) image brute de l'objet | (b) Estimation des flux sur (a)

(c) soustraction du fond de 1.6-(f) | (d) soustraction du fond (c) sur (b)

(e) Image dé-biaisée issue de (d) | (f) Résultat final

FIG. 1.6 – Etapes intermédiaires de la procédure Image-Cell

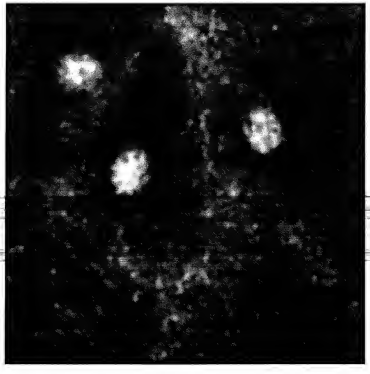


FIG. 1.7 – Résultat final après reconstruction (cetteimage a juste été lissée, et n'est pas une reconstruction par RBF).

**Supprimer le zoom sur l'image :** le zoom est utile pour diminuer l'effet de l'ouverture numérique avant la LPE. Il faut trouver l'opérateur équivalent sur l'image non zoomée. Une hypothèse à creuser est que ce serait équivalent à une ouverture avec un élément structurant en croix (?).

**Auto-contrôle de la détection des fibres :** savoir si la détection est correcte, ou si au cours des acquisition on arrive à un moment où il faut la refaire.

**Auto-contrôle de valeur du fond :** savoir sir le fond proposé est valable ou si il faut l'estimer. Dans le cas où le fond n'est pas fournit, en faut-il un ?

**Amélioration continue de la détection des fibres :** on devrait pouvoir utiliser les images des objets pour renforcer ou corriger la détection des fibres.

**Estimer le biais avec des splines :** le biais est estimé par rééchantillonnage bilinéaire qui produit un effet de blocs qui peut être gênant

On peut aussi prévoir plusieurs axes de recherche pour des fonctionnalités futures : ce sera le sujet d'un prochain mémo...

# Chapitre 2

# Programmation d'Image-Cell

## Introduction

**Auteur CVS (du chapitre) :** *Author* : *aymeric*

**Version (du chapitre) :** *Revision* : 1.2

**Référence du document :** `$Id: chapInfo.tex,v 1.2 2002-04-24 07:14:48 aymeric Exp $`

Cette partie rassemble les principaux éléments de la librairie. La description de ces élément est globale, cette documentation venant en complément de la documentation automatique[1] décrivant toutes les classes et toutes les fonctions, et des tests unitaires décrivant l'utilisation des classes et fonctions principales.

## 2.1 Une librairie sur les graphes

La programmation des algorithmes a nécessité l'utilisation de structure de graphes. Nous avons donc recherché des librairies C++ permettant de les manipuler. Nous avons testé les librairies suivantes.

**GTL :** pour Graph Template Library (`http://www.infosun.fmi.uni-passau.de/GTL/`). Cette librairie est la plus adaptée à nos besoins et a été celle utilisée pour le prototypage dans [?]. Le problème principal de cette librairie est son prix dissuasif pour les entreprises : 5 000 Euros (licence de développement pour un seul poste). Elle était gratuite pour le stage car utilisée par une étudiante... Nous n'avons donc pas choisi de garder cette librairie pour cette raison.

**LEDA :** pour Library of Efficient Data Types and Algorithms (`http://www.mpi-sb.mpg.de/LEDA/`). Cette librairie traite de nombreux problème de type de données, dont celui des graphes. Cette librairie a cependant beaucoup évoluée vers une sorte de librairie couteau suisse qui fait beaucoup de choses, dont les interfaces graphiques, et qui redéfinit de nombreux outils maintenant standard, notamment les types STL. Il existe donc de nombreux cas de conflits potentiels avec nos softs. Le prix de cette librairie est de 1500 US$ pour un développeur, et 8600$ pour développer sur tout un site, et vendre des produits derrière. J'ai personnellement utilisé cette librairie pour ma thèse (gratuitement car pour une thèse...).

**BOOST :** une sorte de groupware qui travail sur des proposition de futurs standard C++ (`http://www.boost.org/index.htm`). Il y a en fait toute un librairie avec de nombreux thèmes. Le point central est que c'est de la programmation avancée, générique, réutilisable, versatile. C'est beau mais c'est très compliqué à utiliser malgré une documentation très

[1] Utilisant doc++

propre. Le problème vient des pré-requis indispensables en programmation objet de haute voltige qui sont un peu trop complexes pour nos besoins, et qui nécessiterai un travail supplémentaire important. Cette librairie est cependant gratuite, et très bien faite. Un attrait intéressant est qu'elle n'est constituée que de fichiers .h, sans partie à compiler. Tout repose sur une gestion complexe de templates qui ne peuvent pas être précompilée. Il n'y a donc pas de problèmes de compilation sur différentes plateformes.

Toutes ces librairies fonctionnent sous toutes les plateformes courantes, et avec tous les compilateurs courant. De plus elles sont maintenues et stables. Il existe de nombreuses autres petite libriaires à droite à gauche mais qui sont loin d'approcher la qualité de celles-ci. Nous n'avons cependant pas pu en retenir une celle, soit à cause du prix et des risques liés à l'utilisation d'une grosse librairies supplémentaire, soit à cause de la complexité d'utilisation trop grande.
Nous avons donc choisi de prendre nos propres structures de données, adaptés aux traitement pour plus de rapidité à la fois de développement et d'exécution. Ces structures de données sont cependant assez souples car elle fonctionnent sur des types et des algorithmes STL qui peuvent être interchangeables assez rapidement...

## 2.2 Gestion générale des paramètres d'algorithmes

### 2.2.1 L'objet paramètre

L'objet paramètre n'existe pas encore en tant que tel, mais va bientôt voir le jour... Il est pour le moment à l'état de concept dans le même sens que la terminologie STL.
Une classe paramètre est de type suivant :

```
class AlgoParam {

  public :

  AdjGraphParam(){
    defaults();
  };

  void defaults() { // Cette fonction fixe les paramètres par défaut

  }

  /// Load parameters from a file. If any error occurs, default values are provided.
  void loadParam(const char * fname);

  /// Save parameters in a file.
  void saveParam(const char * fname);

  /// Output stream
  friend std::ostream & operator << (std::ostream &, AlgoParam &);

  /// Input stream
  friend std::istream & operator >> (std::istream &, AlgoParam &);

public: // parameters list here, with comments please !

[...]
```

};

Les fonctions loadParam et saveParam vont juste créer un flot dans le fichier dont le nom est spécifié. Ils déléguent donc le travaille aux flots `ostream` et `istream`.
Pour le moments les flots sont simples, et ne font que écrire le nom du paramètre suivie de la valeur numérique, le tout sur une ligne. Les paramètres sont encadrés par

```
ALGOPARAM_CONFIGURATION
PARAMETERS
```

et

```
END_ALGOPARAM_CONFIGURATION
```

Ce mécanisme (surement perfectible...) permet de déléguer les entrées et les sorties aux flots qui peuvent donc être imbriqués. Ainsi un objet paramètre peut contenir un autre objet paramètre sans avoir à récrire les entrées et les sorties de cet objet.
De telles classes de paramètres sont utilisés pour `ImageCell` `FiberDetector` `AdjGraph` et `FiberFlow`.

### 2.2.2 Le futur des paramètres

La première chose à faire est de créer une classe virtuelle (c'est-à-dire une interface) pour la classe de paramètres.
Enfin, dans un second temps est à l'étude l'utilisation d'XML pour interfacer ces paramètres.

## 2.3 La détection des fibres : FiberDetector et AdjGraph

Voir le test unitaire `test_fiberdetect` pour les détails d'utilisation.
L'objet FiberDetector est un objet-algorithme modulaire pour détecter les fibres. Il contient un objet spécialisé AdjGraph qui permet de faire l'étape de fusion des fibres qui est la plus compliquée.

### 2.3.1 Fonctionnement détaillé

On retrouve les étapes suivantes dans la classe :

```
void init(Inrimage *inrIn);
void preprocessing();
void watershed();
int splitBiggest ();
int mergeSmallest ();
```

L'étape d'initialisation `init` permet de fixer l'image servant de base à la détection des fibres. Puis l'étape `preprocessing` fait es premiers traitements : zoom, ouverture numérique, inversion de l'image, et quelques calculs préliminaire, comme une structure de voisinage des fibres. Ces deux étapes permettent de préparer l'image pour les traitements de segmentation.
La segmentation est réalisée par une ligne de partage des eaux (LPE) dans la fonction `watershed`. L'algorithme pourrait également interfacer un autre algorithme...
Les deux étapes suivantes sont deux étapes de post-processing de *split and merge*. `splitBiggest` va repérer les fibres trop grosses, puis les resegmenter. `mergeSmallest` permet de fusionner les fibres les plus petites. Cette dernière étape plus complexe et plus critique est elle-même encapsulée dans une classe algorithme AdjGraph expliquée dans la prochaine partie.
Ces deux étapes contiennent une dizaine de paramètres qui permettent d'ajuster la précision et la sensibilité de l'algorithme. La structure de l'algorithme permet d'éventuellement boucler ces étapes afin de faire converger l'algorithme vers une segmentation stable au sens de l'idempotence du *split and merge*, pour un jeu de paramètre donné[2].

[2]Nous n'avons pas de preuve de l'existence d'une telle convergence et il semble difficile *a priori* d'en obtenir.

### 2.3.2 L'étape de fusion avec AdjGraph

Cette partie décrit l'étape de fusion. Cette étape est importante car les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final. La complexité de cette partie nos a obligé à en faire une partie modulaire, au même titre que le `FiberDetector`. Les différentes «modules» ont été les suivants :
- type de calcul de la compacité d'une région, et de la fusion de deux régions
- construction du graphe d'adjacence dédié
- structures de données dédiée aux calculs des attributs utiles.

Cette étape était la plus longue sur le prototype de [?], et a donc fait l'objet d'optimisation. L'étape de fusion n'est réalisé que sur des critères de formes, sans lien directe nécessaire avec l'image de départ[3]. Le problème principal est le calcul des attributs utilisés : la compacité et la surface. La principale difficulté est liée à ce que le graphe va évoluer par fusion de régions, et que l'on cherche à mettre à jour ces attributs ; on veut également tester les fusions possibles et donc pré-calculer les attributs possibles.
Pour cela, on distingue les frontières extérieures et intérieures d'un objet comme indiqué sur la figure 2.1. On divise la frontière d'un objet en fonction de l'adjacence. La frontière de A est égale à la somme des frontière intérieures de A avec tous ses voisins. La frontière intérieure de A commune avec F est égale à la frontière extérieure de F commune avec A. Cette dernière propriété permet de ne parler que de frontière intérieure.

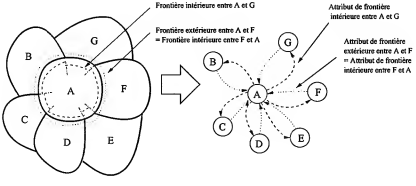



FIG. 2.1 – Les frontières extérieurs et intérieures d'un objet. La frontière intérieure de A commune avec F est égale à la frontière extérieure de F commune avec A. Ces informations sont portés par des arcs du graphe d'adjacence.

Ces informations sont portés par des arcs du graphe d'adjacence. Un arc allant de A vers B va porter l'information de frontière commue que A détient avec B.
On peut formaliser ces relations ainsi. Soit $G = (N, E)$ un graphe avec $N$ l'ensemble de ses noeuds et $E \subseteq N \times N$ l'ensemble de ses arcs. G est un digraphe[4] où deux nœuds n'ont soit pas d'arcs en commun, soit deux arcs antisymétriques. $E$ est donc un ensemble de paires ordonnées. Soit $s$ : $N \rightarrow I\!R^+$ une fonction permettant d'obtenir la surface. Soit $f : E \rightarrow I\!R^+$ une fonction permettant d'obtenir la frontière intérieure entre les deux nœ désigné par l'arc. Enfin soit $c : N \rightarrow I\!R^+$ la fonction permettant d'obtenir la compacité d'une région. On note $\mathcal{V}(n)$, $n \in N$ l'ensemble des voisins de $n$. La compacité peut alors s'écrire :

$$c(n) = \frac{\left(\sum_{m \in \mathcal{V}(n)} f((n,m))\right)^2}{s(n)}. \tag{2.1}$$

[3]Ce qui est en fait une extension (lourde !) possible
[4]un graphe où tous les arcs sont dirigés

Pour tester la compacité de la fusion éventuelle de $n$ et $m$, on note la nouvelle compacité $c(n \cup m)$ :

$$c(n \cup m) = \frac{\left(\sum_{o \in \mathcal{V}(n)} f((n,o)) + \sum_{o \in \mathcal{V}(m)} f((m,o)) - f((n,m)) - f((m,n))\right)^2}{s(n) + s(m)}. \quad (2.2)$$

Et si la fusion de $m$ avec $n$ est effectuée, on modifie les fonctions (on suppose que $m$ est rajoutée à $n$ ) :

$$s(n) \leftarrow s(n) + s(m) \quad (2.3)$$

$$f(n,o) \leftarrow f(n,o) \text{ pour } o \in \mathcal{V}(n) \text{ et } o \notin \mathcal{V}(m) \quad (2.4)$$

$$f(n,o) \leftarrow f(n,o) + f(m,o) \text{ pour } o \in \mathcal{V}(n) \cap \mathcal{V}(m). \quad (2.5)$$

On peut ainsi ne travailler que sur les graphes, sans avoir à revenir à la segmentation initiale pour recalculer les attributs lors d'une fusion. Cela nous a permis de tester beaucoup plus de fusion, voire éventuellement de tester toutes les fusions possibles.

## 2.4 Structures de données associées aux fibres : FiberStructure et FiberInfo

Voir le test unitaire `test_fiberInfo` pour les détails d'utilisation.

### 2.4.1 Lien entre la structure des fibres et les informations sur chaque fibre

La structure de donnée est spécifique au problème qui nous intéresse. Les informations qui nous intéressent sont des informations attachées aux fibres seulement, les fibres étant en fait liées à la structure détectée. Les deux types d'informations sont donc étroitement liées, les informations sur les fibres ne pouvant exister indépendamment de la structure des fibres.

#### Modèle sujet-observateur

Nous avons donc choisi de les relier sous le modèle de l'observateur des design patterns [?]. Nous avons donc défini ce mécanisme dans le fichier `Observer.h`. Le principe est le suivant :

> «Définit une interdépendance un à plusieurs, de façon telle que, quand un objet change d'état, tous ceux qui en dépendent en soient notifiés et automatiquement mis à jour.»

Ainsi on distingue le sujet, qui pour nous est la structure des fibres détenue par la classe `FiberStructure` ; et les observateurs qui sont les informations que l'on peut attacher aux fibres (classe `FiberInfo<T>`).
On a ajouté un comportement supplémentaire : un objet `FiberInfo<T>` ne peut exister sans un `FiberStructure`, ce pourquoi il n'existe qu'un constructeur qui prend en argument la `FiberStructure`.
L'objet `FiberInfo<T>` va d'ailleurs s'initialiser tout seul en regardant la structure à laquelle il est attaché.
Quand la structure vient à changer, tous les `FiberInfo<T>` en sont informés si besoin, et ceux-ci vont alors se réinitialiser automatiquement. Quand la structure est détruite, les objets d'information sont informés également qu'ils ne sont plus valides.

#### Structure codée comme un graphe : FiberStructure

La classe fiberstructure se construit à partir de l'image des composantes connexes. L'objet va alors garder en interne une copie de cette image de référence. Elle contient en interne la structure de graphe associée alors aux composantes connexes. Elle contient également d'autres informations, comme les barycentres des régions, en coordonnées pixel, les surface des régions ...

Toutes ces informations sont accessibles depuis l'interface de la classe, mais ne sont pas modifiables, car elles dépendent de la structure même.

#### La classe d'information : FiberInfo

La classe `FiberInfo<T>` est une classe template car les informations peuvent être de nature très différentes. Elle contient un vecteur STL d'information dont la taille est gérée automatiquement en fonction de la structure associée. Ce vecteur est un membre *public* afin d'avoir des accès rapides aux informations. La classe sert juste de décorateur[5] pour pouvoir gérer les dépendances avec la structure des fibres.
Quand le type du template est numérique, il existe des fonctions pour additionner, soustraire, diviser et multiplier deux `FiberInfo<T>`, ou bien un `FiberInfo<T>` avec un élément de `T` : `addFI`, `subFI`, `divFI`, `mulFI`.

### 2.4.2 Transformation d'images en FiberInfo et réciproquement

Voir le test unitaire `test_fiberProcessing` pour les détails d'utilisation.
Une image peut être transformée en `FiberInfo<float>` avec la fonction d'estimation simple des flux :
`int flowEstimation (Inrimage* image, FiberInfo<float> & flow, bool bias = false)`.
Il faut au préalable initialiser de `FiberInfo<float>` `flow` avec la structure, ainsi l'algorithme connaît via `flow` l'image des composantes connexes à utiliser.
La transformation réciproque est la reconstruction d'image, pour laquelle il n'existe pour le moment que la méthode de mosaïquage :
`int imageMosaic (Inrimage* image, FiberInfo<float> & flow)`
Le principe est similaire à `flowEstimation`. La structure est passée via le `flow`.

## 2.5 Soustraction du fond et biais : FiberFlow

Voir le test unitaire `test_fiberFlow` pour les détails d'utilisation.

### 2.5.1 Fonctionnement général

La classe FiberFlow implémente le bloc d'estimation des flux de la partie 1.1, sauf la partie de transformation de l'image en `FiberInfo`. La classe comporte deux méthodes principales : `setBackground` et `setImage`, et son comportement est paramétré avec la classe de paramètre `FiberFlowParam`. La classe permet de soustraire le fond qui est donnée, ou bien il peut l'estimer. La classe sert aussi à calculer le biais dans l'image, après soustraction du fond.

### 2.5.2 Le calcul du fond

Voir le test unitaire `test_fiberHisto` pour les détails d'utilisation.
Pour calculer le fond, on calcul un quantile de l'histogramme des fibres. Pour cela il existe une classe `FiberHisto<T>` qui prend un `FiberInfo<T>` en constructeur, et qui permet de faire des calculs sur l'histogramme calculé à partir de l'information des fibres. Ces calculs sont plus rapides que sur l'image elle-même.

[5] Attention, pas au sens du Design Pattern décorateur [?]!

### 2.5.3 Le calcul du biais

Voir le test unitaire `test_fiberProcessing` pour les détails d'utilisation.
Le calcul du biais n'est en fait pas calculé sur l'image, mais encore une fois sur l'information des fibres directement. C'est encore un cas où on utilise la structure et l'information des fibres. Ce calcul est fait dans une fonction :

```
int biasEstimate  ( FiberInfo<float> & flowIn,
                    FiberInfo<float> & bias,
                    int nbBlocks,
                    biasEstimatorType bet=BET_MEAN)
```

L'estimateur du biais est pour le moment à choisir entre la moyenne (`BET_MEAN`) et la médiane (`BET_MEDIAN`).
On divise l'espace total des fibres en `nbBlocks` blocs de taille égale. On parcourt ensuite l'ensemble des blocs et on trie les fibres en fonctions du bloc dans lequel elles tombent. Puis, pour chaque bloc, on applique l'opérateur choisi d'estimation du biais.
On calcule le biais ensuite en interpolant pour chaque fibre uniquement (et non pour toute l'image) les valeurs des blocs les plus proches. On utilise pour le moment une interpolation bilinéaire, car c'est la seule disponible avec Inrimage. On fait une proposition pour avoir une interpolation spline cubique.

## 2.6 Améliorations

- La classe de paramètres est à optimiser et à programmer plus proprement.
- Le biais devrait être interpolé avec une spline cubique
- Une librairie de graphe serait tout de même la bienvenue. Cela permettrait d'avoir une structure commune entre `FiberDetector`, `AdjGraph`, et `FiberStructure`, ce qui n'est actuellement pas exactement le cas.

# Chapitre 3

# Utilisation d'Image-Cell

**Auteur CVS (du chapitre) :** *Author : aymeric*

**Version (du chapitre) :** *Revision : 1.2*

**Référence du document :** `$Id: chapUtil.tex,v 1.2 2002-04-24 07:14:48 aymeric Exp $`

C'est une partie délicate car elle doit contenir le savoir-faire acquis pendant le prototypage, le développement et les nombreux tests.
Les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final.

## 3.1 Utilisation de la paramétrisation

Il faut ouvrir une fenêtre d'un éditeur de texte (notepad, emacs, wordpad...) avec le fichier imagecell.txt. Ce fichier est relu à chaque utilisation. La liste des paramètre, ainsi que des résumés d'indications sont données en annexe A.

## 3.2 Paramétrage de FiberDetector

### 3.2.1 Prétraitements

Dans presque tous les cas, il faut faire le zoom sur l'image pour que l'algorithme fonctionne correctement.

### 3.2.2 Paramétrage de AdjGraph

Les petites fibres (sur-segmentées) sont plus gênantes que les grosses (sous-segmentées) sur un résultat final.

## 3.3 Paramétrage du FiberFlow pour la calibration

## 3.4 Paramétrage du FiberFlow pour l'image à reconstruire

## 3.5 Paramétrage d'ImageCell : les reste des paramètres

Annexe A

# Paramètres de Image-Cell

Les paramètres internes d'Image-Cell sont modifiables via un fichier de paramètres présent dans le répertoire source du lancement de l'application, au même endroit que tomoscope.cfg. Le fichier des paramètres `imagecell.cfg` a la forme suivante :

| Ligne du fichier | Défaut | Possibles | Remarques |
|---|---|---|---|
| IMAGECELLPARAM_CONFIGURATION | | | Paramètres de l'algorithme entier |
| PARAMETERS | | | - |
| FIBERDETECTORPARAM_CONFIGURATION | | | Paramètres de l'algorithme entier |
| PARAMETERS | | | - |
| _doZoom_ | 1 | $\{0,1\}$ | Travail sur l'image doublée (plus précis) |
| _doDiffusion_ | 0 | $\{0,1\}$ | Option non pris en compte pour le moment. A régler. |
| _hDomeWatershed_ | 1 | $\{1,..,10\}$ | Hauteur des maxima sélectionnés pour initialiser les fibres. |
| _oversizedFactor_ | 1.8 | $[1;2]$ | Taille normalisée (par rapport à la moyenne) au delà de laquelle une fibre va être divisée. Ce critère est utilisé avec le critère suivant sur la taille (ET logique) |
| _overSizedNeighLimit_ | 6 | $\{5,6,7\}$ | Nombre de voisins au delà duquel une fibre va être divisée. Ce critère est utilisé avec le critère précédent sur la taille (ET logique) |
| _overSizedResegmentMethod_ | 1 | $\{0,1\}$ | 0 : resegmenter dans une fonction distance ; 1 : resegmenter dans l'image originale sans prétraitement |
| ADJGRAPHPARAM_CONFIGURATION | | | Paramètres pour la fusion des fibres |
| PARAMETERS | | | - |
| _underSizedFactor_ | 0.6 | $[0.1;1]$ | Taille normalisée (par rapport à la moyenne) en deçà de laquelle une fibre va être une candidate pour la fusion. Ce critère est utilisé avec le critère suivant sur la taille (ET logique) |
| _underSizedMendatoryMergeSlope_ | $-0.22$ | $[?;?]$ | Ce paramètre et le suivant déterminent une droite dans l'espace (nombre de voisins)x(taille normalisée) délimitant les fibres qui doivent obligatoirement être fusionnées. *Mieux vaut ne pas y toucher sans faire de calcul précis !* La droite par défaut passe par les points $(6;1/3)$ et $(3,1)$. Augmenter la pente revient à s'occuper plus des grosses fibres avec peu de voisins. |
| _underSizedMendatoryMergeOffset_ | 1.667 | $[?;?]$ | Voir remarque précédente. Ce paramètre est l'ordonnée à l'origine (axe des ordonnées=tailles normalisées). L'augmenter revient à obligatoirement fusionner plus de fibres. |
| _underSizedFilterSize_ | 1 | $\{0,1\}$ | Doit-on filter les fusions possibles en fonction de la taille totale de la fusion ? |
| _underSizedFilterSizeMax_ | 1.17741 | $[1;3]$ | Si oui, interdire les fusions donnant une taille de fibre normalisée supérieure à |
| _underSizedFilterCompact_ | 0 | $\{0,1\}$ | Filtrage des fibres fusionnées en fonction de la compacité |
| _underSizedFilterCompactMax_ | 2 | $[1;10]$ | Facteur multiplicatif de l'écart-type |

| Ligne du fichier | Défaut | Possibles | Remarques |
|---|---|---|---|
| _underSizedFilterMoreCompact_ | 0 | $\{0,1\}$ | Ne garde que les fusions qui rendent les fibres fusionnées plus compacte que la fibre voisine seule. |
| END_ADJGRAPHPARAM_CONFIGURATION | | | |
| END_FIBERDETECTORPARAM_CONFIGURATION | | | |
| FIBERFLOWPARAM_CONFIGURATION | | | Paramètres d'estimation du flux pour l'image de calibration |
| PARAMETERS | | | - |
| _backgroundMode_ | 1 | $\{0,1,2\}$ | 0 : pas de soustraction du fond ; 1 : soustraction avec le fond fourni, sinon 2 ; 2 : estimation du fond sur l'histogramme |
| _backgroundEstimationCut_ | 0.01 | $[0;0.1]$ | Quantile de l'histogramme de l'image pour l'estimation du fond |
| _biasCorrection_ | 1 | $\{0,1\}$ | Correction du biais, après soustarction du fond |
| _biasCorrectionNbBlocks_ | 16 | $\{2,..,64\}$ | Si correction du biais, nombre de blocks en X et en Y pour son estimation |
| _biasCorrectionOperator_ | 1 | $\{0,1\}$ | Opérateur pour la correction du biais : 0 : moyenne ; 1 : médiane |
| END_FIBERFLOWPARAM_CONFIGURATION | | | |
| FIBERFLOWPARAM_CONFIGURATION | | | Paramètres d'estimation du flux pour l'image à reconstruire |
| PARAMETERS | | | - |
| _backgroundMode_ | 1 | $\{0,1,2\}$ | 0 : pas de soustraction du fond ; 1 : soustraction avec le fond fourni, sinon 2 ; 2 : estimation du fond sur l'histogramme |
| _backgroundEstimationCut_ | 0.01 | $[0;0.1]$ | Quantile de l'histogramme de l'image pour l'estimation du fond |
| _biasCorrection_ | 1 | $\{0,1\}$ | Correction du biais, après soustarction du fond |
| _biasCorrectionNbBlocks_ | 4 | $\{2,..,64\}$ | Si correction du biais, nombre de blocks en X et en Y pour son estimation |
| _biasCorrectionOperator_ | 1 | $\{0,1\}$ | Opérateur pour la correction du biais : 0 : moyenne, 1 : médiane |
| END_FIBERFLOWPARAM_CONFIGURATION | | | |
| _tempoFilterTypeFib_ | 3 | $\{0,..,4\}$ | Type du filtre temporel pour l'estimation des fibres :0 : moyenne ;1 : moyenne coupée ; 2 : médianne ; 3 : max ; 4 : min |
| _calibration_ | 1 | $\{0,1\}$ | Effectuer la calibration |
| _outputSigmaFilter_ | 3 | $[0;10] \cup \{-1\}$ | Ecart type de la gaussiène pour un filtrage passe bas en sortie de image cell. Si la valeur est négative (-1), il n'y a pas de filtrage. |
| END_IMAGECELLPARAM_CONFIGURATION | | | |

# Bibliographie

**Diligence Document 8**

| MAUNA KEA TECHNOLOGIES | CAHIER DES CHARGES LOGICIEL PROTO | **Date :** 14 septembre 2001<br>*Revision* : 1.11 |
|---|---|---|
| | | **Rédacteurs :** G. Le Goualher, A. Perchant |
| | | **Diffusion :** Personnel MKT |

## Contenu du document

Ce document est le cahier des charges de l'application logicielle du prototype de biotomoscope. Se référer également au document **Implémentation [?]** présentant l'implémentation logicielle.

Ce document : `$Id: cahierdeschargesPROTO.tex,v 1.11 2001/09/13 16:05:13 georges Exp $`

# Table des matières

# 1 Présentation générale de la section logicielle

## 1.1 Spécifications techniques

La section logicielle du `tomoscope` doit consister à :
- permettre le contrôle d'une partie des composants opto-électroniques ;
- récupérer des données (images 2D) ;
- assurer la gestion, la visualisation, le traitement et l'analyse de ces images.

## 1.2 Solutions possibles

- solution1 : Une solution consiste à construire une application logicielle tournant sur un PC communicant avec le boîtier opto-électronique.
- solution2 : Les concepteurs de systèmes endoscopiques offrent des systèmes embarqués permettant l'acquisition et muni d'une sortie vidéo permettant l'affichage des images sur un moniteur TV standard.

## 1.3 Analyse

La conception d'un système embarqué avec affichage des images sur un écran TV suffit dans un contexte très bien déterminé où l'opérateur n'a pas le temps (ou la possibilité) d'interagir avec une application trop complexe. Le développement d'un tel système peut être mené par des ingénieurs possédant des connaissances à la fois en hardware et en software. Il implique la recherche et la mise en oeuvre de carte électroniques, la programmation sur OS spécialisé généralement bas niveau (assembleur).

La mise en oeuvre d'un système basé sur une composante embarquée (boîtier opto-électonique) connecté à un PC dédié permet à des ingénieurs software de développer une application modulaire facilement modifiable. L'utilisation d'un PC standard, utilisant un OS standard permet de se placer dans une configuration évolutive ou les composants développés par des entreprises tiers peuvent être utilisés. Le travail de prototypage effectué sur un système PC/boîtier opto-électronique peut être transcrit sous la forme d'un système embarqué lorsque les besoins sont bien standardisés.

## 1.4 Choix

Etant dans une phase de prototypage, nous jugeons que la conception d'un système embarqué, peu modulaire ne peut être retenue. Cette solution pourra être étudiée dans une phase futur si le design de l'ensemble du produit s'avère stable. Il est fort probable que la réalisation d'une intégration optimale impliquera un accord avec un constructeur.

Nous optons donc pour une solution composée d'un boîtier opto-électronique connecté à un PC sur lequel tourne une application logiciel dédiée répondant aux services énoncés ci-dessus dans la section `spécifications techniques` (section 1.1) .

# 2 Scénario de mise en oeuvre

## 2.1 Scénario de mise en oeuvre et délais associés

Le scénario de mise en oeuvre du système est constitué des étapes suivantes (lesquelles doivent être réalisées dans l'ordre) :

1. connexion du boîtier opto-électronique vers le PC : connexion série sur le port COM2, et branchement de la carte d'acquisition CNAM embarquée sur le PC avec une nappe au format LVDS ;

2. mise sous tension et le démarrage du boiter opto-électronique ;
3. mise sous tension et le démarrage du PC ;
4. login de l'utilisateur et démarrage de l'application logicielle du tomoscope.

Le délai de mise en oeuvre peut donc être considéré (pour la partie software du moins), comme :

1. temps de boot du PC (si nécessaire) : 3 minutes ;
2. temps de lancement de l'application : 5 secondes ;
3. choix par l'opérateur de la zone disque où vont être stockées les images (cf. module Browser **modBROWSER** section 4.2) : 1 minute ;
4. temps du lancement du module d'acquisition **modBROWSER** (cd. section 4.3) : 5 secondes.

Soit un temps cumulé d'environ 4 minutes si le PC doit être booté ou d'environ 1 minute si le système PC/boîtier opto-électronique est déjà mis en oeuvre.

## 2.2 Configuration matérielle

La configuration du PC est la suivante :
- boîtier rack 19" 4U ;
- écran plat 17" ;
- bras de support écran mobile (2 axes, réglable en hauteur) ;
- processeur PIII $\geq$ 700 Mhz ;
- disque dur de $\geq$ 30 Go 7200 TRs ;
- 256 MO de mémoire SDRAM ;
- lecteur de disquette ;
- lecteur CD ;
- graveur de CD ;
- carte vidéo MATROX G450 16 MO SDRAM dual head (permettant d'utiliser un second ecran) ;
- carte réseau ;
- carte d'acquisition CNAM ;
- clavier industriel (incluant un trackball) ;
- OS : Windows NT 4 service pack 6 ;

Il s'agit donc d'un PC d'un coût approximatif de 12 000 FF (sans la carte d'acquisition CNAM) et d'une configuration complete de l'ordre de 25 000F. Les éléments coûteux étant actuellement le boîtier rack 19", le bras mobile et l'écran plat. L'OS devra évoluer vers Windows2000 et éventuellement Linux.

## 2.3 Intégration hospitalière

Pour la pratique hospitalière on peut envisager de mettre le PC et le boîtier électronique sur une colonne dédiée. L'écran doit être suffisamment proche du praticien. Le point critique est de positionner le connecteur de la sonde de telle façon à ne gêner ni l'utilisateur du tomoscope, ni une éventuelle personne assise utilisant le PC.

# 3 Conception & développement de l'application logicielle

## 3.1 Spécifications techniques

La partie logicielle fournie avec le tomoscope doit :

- offrir un service minimal de gestion des données (images/commentaires) permettant de stocker, visualiser les acquisitions réalisées avec le tomoscope ;
- permettre l'acquisition et la visualisation en temps réel des images lors d'une session d'acquisition dans de bonnes conditions ;
- permettre à l'utilisateur de visualiser et d'effectuer des post-traitements sur les images sauvegardées lors d'une session.

## 3.2 Solutions possibles

- solution 1 : Les besoins spécifiés dans la section précédente étant des besoins relativement standards, la première des solutions consisterait à réutiliser une application existante offrant ces services.

- solution 2 : Recherche d'un toolkit permettant de développer par nous même une telle application.

## 3.3 Analyse

- solution 1 : Trouver une telle application, suffisamment modulable pour y intégrer nos propres spécificités (comme en particulier la communication entre les cartes électroniques et le PC), implique la recherche d'un partenariat (consultant/sous-traitant) avec une entreprise fournissant ce genre d'offre. Cela peut être coûteux et aucun choix évident n'apparaît au niveau auquel nous nous trouvons actuellement. Il est en effet très difficile aujourd'hui de pouvoir déterminer avec certitude les évolutions du soft.

- solution 2 : Plusieurs toolkits existants sur le marché. Parmi les plus connus : les toolkits Microsoft basés sur les MFCs, les produits Borland, des produits gratuits comme tcl/tk. Trolltech `www.trolltech.com` propose Qt un framework jouissant d'une bonne réputation (bien documenté, prise en main facile, écrit en C++). Le grand avantage de Qt par rapport aux autres toolkit est son aspect multi-plateforme (en particulier le code écrit tourne sur Linux et WindowsNT). Qt possède des fonctions pour la localisation (choix de la langue).

## 3.4 Choix

Nous avons opté pour un développement basé sur `Qt (www.trolltech.com)`. Nous essayons de garder le plus possible l'aspect multi-plateforme cependant les modules d'acquisitions et de commande de l'électronique ne fonctionnent pas sous Linux car nous ne disposons pas de drivers pour la carte d'acquisition du CNAM. Nous gardons cependant une compatibilité Linux/WindowsNT pour le reste. Le choix de l'aspect multi-plateforme permet de réduire les coûts de développement en utilisant les produits gratuits existants sous Linux. Le fait de compiler sous deux plateformes différentes permet également de mettre en évidence des bugs.

# 4 Description du logiciel

Les spécifications techniques de la partie précédente nous donne directement le découpage modulaire de l'application : une **Interface Générale** et trois modules principaux :

- **modBROWSER** : module de gestion des données (browser) ;
- **modACQ** : module d'acquisition ;

- **modPROCESS** : un module de visualisation/traitement de l'image;

## 4.1 Interface Générale

L'interface principale contient un browser permettant de gérer le stockage et la prévisualisation des images ainsi que les paramètres et commentaires associées aux images (**modBROWSER**). Elle contient également deux icônes. La première permet d'ouvrir le module d'acquisition (**modACQ**). La seconde permet d'ouvrir le module processing (**modPROCESS**) . La langue utilisée pour l'interface et les commandes est l'anglais.

## 4.2 Module de gestion des données (modBROWSER)

### 4.2.1 Spécifications techniques

Le module de gestion de données a pour objectif de faciliter la tâche de stockage des images acquises ainsi que de leurs paramètres d'acquisitions et les commentaires textes associés. Il ne constitue cependant pas une interface sur une véritable base de données.

### 4.2.2 Solutions possibles

- solution 1 : Utilisation d'une base de données (type Oracle);
- solution 2 : Recherche d'une application existante dans laquelle nous pouvons nous intégrer;
- solution 3 : utilisation d'une arborescence de répertoires classique.

### 4.2.3 Analyse

Un module de gestion de données est généralement composé d'une interface permettant à l'utilisateur d'accéder à un base de données. Les images obtenues avec le tomoscope constituent une partie d'une information qui devra être associée à un patient. Ces images vont être soumises à des contraintes médico-légales. Ces images devront être également accessibles éventuellement par plusieurs personnes (éventuellement hors-site pour une analyse conjointe par un gastroentérologue et un anapath). L'ensemble de ces contraintes et besoins montre l'importance du module de gestion des images. Plusieurs entreprises développent des solutions pertinentes pour ce genre de problèmes (par exemple ETIAM pour les aspects Dicom et la télémédecine, le Pack-PMSI de I-CARE pour la gestion administrative du dossier patient).

Les besoins à l'étape actuelle sont relativement simples : il s'agit d'offrir à l'utilisateur un minimum d'organisation le guidant dans l'archivage des données et la post-analyse de ceux-ci.

### 4.2.4 Choix

Considérant que les besoins à l'étape actuelle sont relativement simples : il s'agit d'offrir à l'utilisateur un minimum d'organisation le guidant dans l'archivage des données et la post-analyse de ceux-ci, nous optons pour la constitution d'un browser sur une hiérarchie de répertoires prédéfinie.

Le browser permet de visualiser une hiérarchie composée de 3 niveaux, nommées par exemple

- Patient : identifiant du patient
- Examen : identifiant de l'examen
- Study : identifiant de l'étude

Les images et commentaires associées obtenues lors de l'acquisition $k$ sont donc stockées dans le répertoire $\texttt{Patient}_p\texttt{/Examen}_e\texttt{/Study}_k$.

Les noms des différents répertoires peuvent être changés, la profondeur de la hiérarchie peut également être modifiée en fonction des besoins.

**Scénario d'utilisation**

Avant de commencer une acquisition, l'utilisateur renseigne les champs mentionnés plus haut. Les images obtenues lors de l'acquisition seront alors sauvegardées dans le répertoire correspondant à ces champs. Les images d'une même session sont donc toutes sauvegardées dans un même répertoire. Les noms des images seront générés automatiquement ou spécifiées par l'opérateur.

**Configuration matérielle**

Les données sont stockées sur le disque dur du PC d'acquisition. Le simple partage du répertoire de stockage permettra un accès commun dans un réseau local.

Nous effectuons une veille pour la recherche de solutions plus complètes pour ce module (jugé peu prioritaire à l'heure actuelle). Une spécification plus complète de la structuration du stockage doit également être réalisée.

## 4.3 Module d'acquisition (modACQ)

### 4.3.1 Besoins

Les différentes concertations avec les experts médicaux, ont conduit aux spécifications suivantes :

- la procédure d'acquisition consiste à poser la tête endoscopique sur la zone d'intérêt, rester immobile à peu près une seconde, déclencher la sauvegarde de l'image;
- le résultat d'une session d'acquisition est un ensemble d'images 2D, chacune des images correspondant à un échantillon sélectionné par le praticien opérateur ;
- la sélection des images se fait par rapport à la qualité diagnostique de celles-ci ;
- il n'est pas nécessaire d'enregistrer en continu les acquisitions faites par le tomoscope ;

**Scénario d'utilisation**

1. les images acquises par le tomoscope défilent sur l'écran (10 images/secondes) ;
2. le praticien pose la tête endoscopique sur une zone d'intérêt ;
3. si le praticien juge que les images qu'il voit sur l'écran sont intéressantes, il commande un arrêt de l'image ;
4. le système dispose alors d'un petit intervalle de temps (maximum 5 secondes) pour appliquer des traitements rapides visant à améliorer la qualité de l'image. Les traitements à appliquer sont décidés par les experts MKT ;
5. l'image de qualité supérieure apparaît à l'écran ;
6. le praticien peut sauvegarder l'image ou décider de continuer ;
7. si le praticien décide de sauvegarder l'image, cette image est stockée automatiquement dans le répertoire approprié. Le praticien (ou son assistant) à la possibilité d'associer quelques commentaires texte à cette image ;

### 4.3.2 Spécifications

A partir des besoins décrits ci-dessus, on peut déduire un ensemble de fonctionnalités pour le module d'acquisition :

1. réception des images depuis la carte d'acquisition du CNAM ;
2. commande de l'électronique CNAM (carte PM, carte de modulation, carte de synchronisation) ;

3. affichage en temps réel des images ;
4. possibilité de bufferiser quelques images ;
5. commande de contrôle de l'affichage/sauvegarde ;
6. traitement temps réel ou quasi temps réel ;

**Réception des images depuis la carte d'acquisition du CNAM :**
Lorsque l'on est dans le module **modACQ** le tomoscope effectue en permanence des acquisitions d'images. Les images sont constituées de mesures de taille 640x640 (TBC), codées sur 8 bits (TBC). Ces mesures ne constituent pas des images isotropes (cf. section 5.2.3). On garantie une acquisition au rythme de 10 images par seconde. Physiquement le transport de données se fait par une nappe au format LVDS connectant le boîtier opto-électronique à la carte d'acquisition CNAM sur bus PCI du PC.

**Commande de l'électronique CNAM :**
Une interface dans le module d'acquisition permet de changer les paramètres de l'électronique (cf. document du CNAM []). Ce contrôle doit être effectué par et sur demande de l'expert MKT uniquement. A terme, ce contrôle sera réservé au mode **expert**. Physiquement on utilise le port série (COM2) et une nappe au format LVDS pour commander respectivement les cartes électroniques et recevoir les images.

**Affichage en temps réel des images :**
On affiche en temps réel les images brutes (c'est à dire sans traitement) 640x640 (TBC), 10 images/seconde (TBC). L'évolution de cet affichage consistera à apporter les traitements temps réel permettant d'améliorer la lisibilité de l'image par le praticien.

**Possibilité de bufferiser quelques images :**
L'expert MKT peut décider d'un nombre d'image que l'on garde en mémoire pour des besoins de traitement éventuel. La modification de ce paramètre est possible lors de la phase de test. A terme, le contrôle de ce paramètre sera réservé au mode **expert**, et préréglé automatiquement.

**Commande de contrôle de l'affichage/sauvegarde :**
Cette section permet d'offrir les services nécessaires aux parties 6 et 7 du scénario d'utilisation (cf. section 4.3.1). L'utilisateur dispose d'une commande déportée (type pédale, joystick, -TBD-) lui permettant de :
- commander l'arrêt sur l'image courante ("toggle pause") ;
- commander la sauvegarde de l'image courante. L'image est alors sauvegardée dans le répertoire spécifié par le module **modBROWSER**.
- lors de la sauvegarde de l'image l'utilisateur (ou son assistant) à la possibilité de sauvegarder un commentaire associé.

**Traitement temps réel/quasi temps réel (TTRQTR)**
- **Spécification technique :** Afficher en temps réel une image de qualité supérieure à celles des images brutes provenant de la carte d'acquisition.

- **Solutions possibles :** Il peut s'agir d'effectuer des traitements purement software comme des manipulations de LUT par exemple permettant d'améliorer la lisibilité des images. L'autre aspect concerne les asservissement de l'électronique à partir de mesures effectuées sur les images acquises visant à optimiser les paramètres de contrôle de l'électronique en fonction de la profondeur par exemple. Ces deux solutions sont bien entendue complémentaire.

- **Analyse :** Les images brutes provenant du tomoscope contiennent un certain nombre de défauts qui peuvent gêner le praticien dans la lecture et l'interprétation de l'image (manque de contraste par exemple, se référer aussi à la section 4.4). De plus l'électronique de balayage peut se dérégler et conduire à des

acquisitions de mauvaises qualitées.

- **Choix :** Le module d'acquisition doit posséder les traitements suivants :
  - **TTRQTR 1** : correction automatique du *gain* : permettant d'ajuster automatiquement le contrôle du gain en fonction de la profondeur. Cette opération s'effectue par analyse de l'histogramme et consiste en une boucle de contrôle entre les images acquises et les cartes électroniques.
  - **TTRQTR 2** : calibration en amplitude. C'est une opération qui implique une phase de calibration effectuée en début d'acquisition et permet de s'affranchir du pattern des fibres (cf. section 4.4).

**Traitement rapide effectué lors d'un arrêt sur image (TREAI)**

- **Spécification technique :** Afficher rapidement (délai inférieure à 5 secondes) une image de qualité supérieure à celle de l'image brute provenant de la carte d'acquisition lors d'un arrêt sur image.
- **Solutions possibles :** Les images brutes provenant du tomoscope contiennent un certain nombre de défauts qui peuvent gêner le praticien dans la lecture et l'interprétation de l'image (cf. section 4.4).
- **Analyse :** cf. section 4.4.
- **Choix :** Le module d'acquisition doit posséder les traitements suivants :
  - TREAI 1 : atténuation du bruit poissonien par moyennage temporel : il s'agit d'une opération qui à pour objectif d'atténuer un bruit intrinsèque à l'acquisition afin de rendre plus facilement lisible l'image. Cette opération implique que plusieurs images soient bufferisées.

**Notes** Ces traitements rapides permettent d'améliorer la qualité de l'image. Ils sont effectués uniquement sur demande de l'expert MKT à partir d'une ou de plusieurs images bufferisées. Lorsque les traitements ne sont pas temps réel, ils interrompent l'acquisition.

## 4.4 Module Processing (modPROCESS)

Ce module permet de visualiser une image stockée dans la hiérarchie et de lui appliquer certains traitements. Ces traitements sont effectués en post-processing, sans contrainte de temps réel, mais avec une contrainte de rapidité suffisante pour ne pas ralentir l'examen. Un certain nombres des modules de traitements présents dans cette section peuvent être utilisés pour les traitements lors d'un arrêt sur image (les TREAIs).

# 5 Traitements des images

Cette partie décrit les différents traitements réalisées sur les acquisitions. Les traitements sont répartis en deux groupes : les Traitements Temps Réels / Quasi Temps Réels (TTRQTR), et les Traitements Effectués lors d'un Arrêt sur Image (TREAI). Ce découpage ne concerne pas directement l'organisation des modules. Les TTRQTR ne sont réalisés que dans le module d'acquisition. Les TREAI peuvent, suivant les besoins, être réalisés dans le module d'acquisition, ou dans la module processing.

## 5.1 Traitements Temps Réels / Quasi Temps Réels (TTRQTR)

### 5.1.1 Correction automatique du gain

**Spécification technique**

Suivant la profondeur et la nature du tissus, la valeur de la puissance reçue sur le détecteur peut varier. La chaîne de détection prévoit plusieurs moyen de contrôler ce niveau lors de sa numérisation. Afin d'être dans des conditions optimales du point de vue image, ce gain doit être contrôlé automatiquement en fonction de l'image acquise.

**Solutions possibles**

Le gain peut être contrôlé via la puissance laser, la carte PM et le gain propre du détecteur (si il est réglable). Il y a deux façons de contrôler le gain : une boucle de rétroaction "semi-aveugle", ou bien en anticipant le résultat sur l'image. La première solution propose juste de diminuer le gain s'il est trop fort, et de l'augmenter si il est trop faible. La seconde solution trouve la bonne valeur du gain par un calcul qui incorpore un modèle numérique de la chaîne d'acquisition et de contrôle.

**Analyse**

La boucle de rétroaction est la solution classique quand une partie du système n'est pas bien connue. La difficulté du contrôle dépend alors de l'ampleur de la non-connaissance de cette partie. Il devient plus difficile de prédire le temps de réponse et la stabilité du contrôle. La seconde solution requiert une bonne connaissance de la chaîne d'acquisition et des moyens de contrôle de cette chaîne. On peut alors anticiper par l'observation de l'histogramme de l'image la valeur correcte du gain à appliquer. Il suffit ensuite de traduire cette valeur du gain en termes de contrôle de la chaîne d'acquisition. Cette dernière méthode à l'avantage de converger en une seule itération de contrôle. Elle sera donc plus rapide dans tous les cas que la boucle de rétroaction.

**Choix**

Le choix se porte sur la seconde solution : par anticipation de la boucle de contrôle. Il faut bien noter que ce choix requiert une bonne connaissance de la chaîne de contrôle.

### 5.1.2 Calibration

**Spécification technique**

Le phénomène d'injection du spot laser dans la fibre combiné à l'échantillonnage des spots fait apparaître le pattern des fibres sur les acquisitions. Une réflexion parasite sur la partie distale du guide va être enregistré par les détecteurs. L'amplitude de chaque point de mesure de l'image est soumis à ces phénomène opto-mécaniques qui va créer une texture gênante pour la lisibilité de l'image. Une procédure de calibration est ici envisagée pour éliminer ces phénomènes.

**Solutions possibles**

Le problème est complexe car on ne peut pas tout éliminer en même temps. Ce que l'on peut faire :

1. soustraire l'image de la réflexion parasite ( obtention de celle-ci quand le guide est hors focus)
2. soustraire l'image d'un milieu diffusant homogène
3. soustraire l'image d'un miroir

**Analyse**

La solution 1 est simple à mettre en œuvre car elle nécessite très peu de manipulations. Elle ne permet que d'enlever un fond sur l'image. Comme cette soustraction se situe après la quantification (8 bits) elle ne permettra en aucun cas de voir des choses que l'on ne pouvait pas voir avant. Cela peut juste aider la visualisation La solution 2 nécessite un contact optimal avec un milieu liquide. Soit par immersion, soit avec une interface. Cette solution permet de retirer le plus de défauts (échantillonnage, guide, système optique...). Par contre sa mise en œuvre est très difficile. La solution 3 suppose un alignement parfait entre le miroir et l'axe optique, ainsi qu'une courbure de champ plate au niveau de la tête endoscopique. On ne connaît pas encore la courbure, et les problèmes d'alignement devraient être réalisés au moyen de pièces mécaniques de haute précision. Les solutions 2 et 3 fond perdre la stérilité de l'appareil, sauf si la mire ou le milieu sont eux-mêmes stériles, donc jetable probablement. La solution 1 est la plus souple. Une combinaison des solutions 1 avec 2 ou 3 est éventuellement possibles. Les solutions 2 et 3 sont plus difficilement compatibles. Quelle que soit la solution, il faut bien voir que le vrai facteur limitant pour nous est la quantification 8bits. Il ne faut donc pas attendre de miracle de la soustraction...

**Choix**

La solution 1 est la moins contraignante. Et en l'absence de miracle on choisit la moins contraignante. (TBC).

### 5.1.3 Calcul automatique du *horizontal time shift*

**Spécification technique**

Le *horizontal time shift* (HTS) est un paramètre qui règle le début de l'échantillonnage des lignes ([?]). Ainsi une mauvaise valeur du HTS va créer un entrelacement des lignes. La valeur de ce paramètre varie en fonction des miroirs et des conditions externes de son mouvement (température, pression...). Ce paramètre doit être réglé automatiquement pour qu'il n'y ait pas d'entrelacement.

**Solutions possibles**

L'entrelacement peut être mesuré par la corrélation des lignes paires avec les lignes impaires.

**Analyse**

La présence des fibres va créer une composante haute fréquence (spatialement...) qui va permettre d'obtenir un pic de corrélation élevé. Le pic de corrélation est unique car le guide est semi-régulier et on ne peut pas se retrouver dans une configuration périodique sur une ligne horizontale.

**Choix**

Le choix est rapide puisqu'il n'y a qu'une seule solution.

## 5.2 Traitements Effectués lors d'un Arrêt sur Image (TREAI)

### 5.2.1 Augmenter le SNR

**Spécification technique**

Les acquisitions sont entachés de plusieurs bruits : le bruit poissonien intrinsèque au comptage de photons, les bruits de la chaîne électronique, de la quantification... Le bruit majoritaire est le bruit poissoniens aux niveaux de puissance où on travaille. Ce bruit gêne la lisibilité de l'image et doit être réduit.

**Solutions possibles**

Les solutions envisagées ne prennent pas en compte les modification possible des éléments optiques et/ou électronique d'acquisition. La réduction du bruit peut se faire de deux façons : soit par un filtrage temporel, soit par un filtrage spatial.

**Analyse**

Le filtrage temporel de type passe-bas permet de rejeter les fréquences hautes qui sont majoritairement du bruit dans le cas d'une image fixe. En l'absence d'information sur les images (bougés,...), on propose un moyennage uniforme. Les traitements spatiaux sont plus complexes et dépendent du contenu de l'image, et de l'information que l'on peut obtenir *a priori* sur ce contenu. Dans notre cas, les hautes fréquences spatiales sont occupées par le bruit, mais aussi par le pattern des fibres. Un filtrage passe-bas spatial rendrait l'image floue, ce qui n'est pas acceptable.

**Choix**

Le choix se porte donc naturellement sur un moyennage temporel uniforme.

### 5.2.2 Suppression du pattern des fibres

**Spécification technique**

Fournir une image non pixélisée : s'affranchir du phénomène d'injection dans les fibres. La lecture de l'image ne doit pas être gênée par l'apparition du guide d'image sur celle-ci.

**Solutions possibles**

Trois solutions sont envisagées. La première est un filtrage spatial de type passe-bas qui utilise la forme fréquentielle des fibres. La seconde est une analyse de la forme des fibres qui permet de rétablir un niveau d'injection constant sur toute la fibre. La troisième solution est une calibration sur une mire uniforme connue qui permet de connaître, pour chaque point de mesure, son atténuation globale dans la chaîne d'acquisition.

**Analyse**

La première solution a l'avantage de pouvoir être implantée rapidement à l'aide de filtres récursifs séparables pour un coût de 15 opérations par pixels, c'est à dire presque en temps réel. De plus les paramètres du filtres sont déterminés automatiquement en fonction des spécifications techniques du guide d'image. Le traitement fait bien disparaître le pattern, mais fait apparaître un artefact réparti sous la forme d'un bruit moyenne fréquence isotrope visible dans les zones homogènes. Ce filtre est déjà contenu dans la bibliothèque inrimage.

La seconde solution propose un résultat différent. Le principe est d'isoler chaque fibre à l'aide de techniques de morphologie mathématique. Cette technique utilise la forme de la fonction d'injection 2D, ainsi que la réflexion parasite en bout de guide. On peut ainsi isoler sans calibration et pour chaque image individuelle, chaque fibre du guide d'image. L'étape suivante consiste à rétablir un niveau constant dans chaque fibre avec une analyse statistique locale. Le temps de traitement envisagé pour chaque image (512x512) est d'an plus 0.5 secondes. L'image résultat est une sorte de mosaïque. Le coût de développement est d'environ une semaine.

La troisième solution permet de compenser globalement l'atténuation de chaque point de mesure. Une hypothèse forte doit être vérifiée : le balayage et l'échantillonnage doivent être très stable (inférieure à $500nm$). Dans ce cas, on peut corriger chaque point de mesure à partir des données calibrée. L'image résultat est proche d'une image sans guide d'image, car on simule une injection presque parfaite en chaque point de mesure. Il peut rester un bruit de quantification pour des points où l'injection était mauvaise. Cette méthode nécessite très peu d'opération par pixel, et un seul scan de l'image (donc temps réel). Cette solution a déjà été développée dans le simulateur

Les deux premières solutions ont l'avantage de n'être que logicielles. La troisième nécessite une manipulation de calibration avant l'examen. Sous réserve d'une très bonne stabilité du balayage/échantillonnage, et d'une bonne injection sur l'ensemble du guide, la troisième méthode devrait fournir de très hons résultats. Cette méthode doit cependant être sensible aux hypothèses de stahilité. Des critères impératifs de robustesse demanderait alors plutôt une des deux premières méthodes (TBC).

**Choix**

TBD. De nombreux tests et l'avis de plusieurs médecins seront nécessaires. Notamment la stabilité du halayage doit être vérifiée sur une échelle de temps importante (de l'ordre de 2h00) et en situation réelle pour la calibration systématique. La troisième méthode semble la plus robuste et est le choix par défaut.

### 5.2.3 Correction des distortions géométriques

**Spécification technique**

La forme spécifique du balayage comhiné à l'échantillonnage ainsi qu'à la chaîne de traitement optique et électronique induit des distortions non linéaires de l'image. Ces distortions doivent être corrigées afin de présenter une image non déformée (espace image isotrope).

**Solutions possibles**

Deux solutions sont envisagées. La première est une modélisation analytique de la chaîne optique-électronique. La seconde solution est une estimation globale des distortions à partir d'une mire (observation d'une grille de taille connue). Ces modélisations permettent de déduire la transformation et de réaliser un ré-échantillonnage bilinéaire de l'image dans une grille pixel carrée connue.

**Analyse**

La première solution (analytique) est simple à mettre en œuvre car la forme théorique du balayage et celle

de l'échantillonnage temporel sont bien connues. Cette méthode ne demande pas de calibration. Le résultat n'est cependant pas garanti car on ne peut pas prendre en compte toutes les distortions de la chaîne.
La seconde solution demande une calibration initiale de l'appareil, et une vérification périodique (TBC). On image une grille connue, et on effectue le recalage de toutes les intersections de la grille par rapport à un modèle analytique de cette grille. Cela permet de quantifier le déplacement de chacun de ces points et d'interpoler une déformation entre ces points. La distortion étant a priori spatialement lentement variable, son évaluation en différents points d'une grille de calibration doit donner un très bon résultat. La calibration devrait de plus être entièrement automatique (TBC). Cette distortion est estimée une fois pour toute pour un jeu de paramètres donné, et non à chaque examen.

**Choix**

TBD. La solution de calibration sur une grille devrait être la meilleure et devra au moins être réalisée en laboratoire pour confirmer le modèle théorique.

### 5.2.4 Visualisation pour le diagnostic

**Spécification technique**

L'image doit être présentée de façon à ne pas présenter d'artefacts de traitement (ou à défaut ils doivent être repérés), et de façon à faciliter l'interprétation.

**Solutions possibles**

Les solutions concernent la visualisation de l'image déjà acquise. Certaines solutions classiques sont empruntées au monde de la radiologie conventionnelle numérique. Les traitements envisagées sont : fausses couleurs (proches des colorants classiques utilisés par les anathomo-pathologistes), fenêtrage linéaire et logarithmique des niveaux de gris, zoom numérique.

**Analyse**

Les fausses couleurs permettent selon l'observateur et les conditions de luminosité de la pièce, de se placer dans des conditions connues d'observation. Le fenêtrage linéaire permet de modifier le contraste de l'image en fonction de la plage de niveau de gris. Le fenêtrage logarithmique permet de privilégier le contraste des zones sombres par rapport à celui des zones claires. Sur une image de type gradient (comme les nôtres), cela permet de faire attention aux zones de transition faibles, par rapport aux zones de transition fortes (qui sont des informations évidentes ***a priori*** pour le praticien (TBC). Enfin, le zoom numérique permet de grossir certaines zones d'intérêt.
Ces traitements de visualisation sont faciles à mettre en œuvre et sont des outils largement répandus. De plus ils impliquent des traitements simples qui ne créent pas d'artefact.

**Choix**

Tous ces outils sont implantés, mais leur utilisation est optionnelle. Un mode de visualisation par défaut est proposé.

**Diligence Document 9**

# INTERFACE *stand-alone* POUR IMAGECELL

Aymeric Perchant

Date : 4 novembre 2002

**Diffusion :** interne, image
**Section :** informatique, tdi
**Sujet :** construction d'une interface Qt pour pouvoir utiliser facilement imagecell.
**Version :** *Revision* : 1.1
**Référence du document :** `$Id: imageCellSA_cdc.tex,v 1.1 2002/11/04 13:24:19 aymeric Exp $`

## 1 Introduction

Ce document est un mini cahier des charges pour la construction d'une interface Qt pour pouvoir utiliser facilement imagecell. Le but est que plusieurs personnes de MKT puissent traiter des images de façon autonome. Le fonctionnement standard d'imageCell ne demande que peu de changements, et qui sont faciles à maîtriser.

## 2 Interface texte existante

Il existe deux versions d'ImageCell : une version pour reconstruire des images, et une autre pour les films.

### 2.1 ImageCell, images fixes

L'application est dans `SoftProj/testApp/imageCell/imageCell.cpp`.

```
Usage : imageCell -i <image> [-b <image>] -f <image> | -g <image> [-c <image>]
                  [-o <image>] [-u <image>] [-p <file>] [-d <file>]
-i  <image> image
-b <image>  background image
-f  <image> fiber image
-g  <image> fiber image component
-c <image>  background fiber image
-o <image>  output image (default imageCell_out.inr)
-u <image>  output fiber image (default: no output for fiber image)
-m <image>  output fiber component image (default: no output for fiber image)
-p  <file>  parameter file
-d  <file>  dump point coordinate and flow into file
```

**Entrées :** (`-i, -b, -f, -g, -c, -p`) on rentre l'image (accompagnée d'un fond ou non), l'image des fibres (accompagnée d'un fond ou non également), et un fichier de paramètres. L'image des fibres peut prendre deux formes : soit la forme brute (`-f, -c`), soit l'image des composantes connexes des fibres (`-g`).

Sorties : (`-o`, `-u`, `-m`, `-d`) on a au moins la sortie de l'image courante (`-o`). On peut également sortir l'image des taux d'injection (`-u`), l'image des composantes connexes (`-m`), l'image des points (`-d`) : on a pour chaque fibre, les coordonnées x,y,z (coord. axiale = la tranche temporelle), et le flux mesuré, soit un quadruplet par fibre, à raison d'une fibre par ligne en ASCII.

L'image de sortie est 2D et 8 bits. L'avantage des options `-m` et `-g` est la rapidité de traitement : cela évite d'avoir à recalculer la détection des fibres plusieurs fois.

### 2.2 ImageCellMovie, films

L'application est dans `SoftProj/testApp/imageCellMovie/imageCellMovie.cpp`.

```
Usage : imageCellMovie -i <image> [-b <image>] -f <image> [-c <image>]
                       [-o <image>] [-u <image>] [-p <file>] [-d <file>]
-i  <image> image
-b <image>  background image
-f  <image> fiber image
-g  <image> fiber image component
-c <image>  background fiber image
-o <image>  output image (default imageCellMovie_out.inr)
-u <image>  output fiber image (default: no output for fiber image)
-m <image>  output fiber component image (default: no output for fiber image)
-p  <file>  parameter file
-d  <file>  dump point coordinate and flow into file
```

C'est la même interface que précédemment, sauf que la sortie est 3D.

## 3 Interface future

### 3.1 Généralités

L'interface doit permettre d'utiliser les fonctionnalités de l'interface en ligne de commande, avec en plus la possibilité de visualiser les images qui rentrent dans le calcul. Les paramètres doivent également être modifiables et récupérables. Pour connaître leurs rôles, on s'appuie sur [1], dernière partie. Le même document explique également l'implémentation d'imageCell.
La seule différence entre imageCell et imageCellMovie est au niveau de la sortie qui est 2D ou 3D. On peut donc intégrer les deux dans la même interface, même si la réalisation utilisera deux objets distincts imageCell et imageCellMovie.
L'utilisation des objets est mise en œuvre dans les deux programmes en ligne de commande, et il suffit de s'appuyer dessus...

### 3.2 Paramètres

Parmi les paramètres, certains sont plus importants que d'autres :

**_doDiffusion_** : préfiltrage pour réduire le bruit (pas encore implanté, mais à prévoir),

**_biasCorrection_** : correction du biais (en double : pour l'image des fibres, et pour l'image à reconstruire)

**_biasCorrectionNbBlocks_** : finesse de la correction du biais (si besoin) : entre 2 et 20.

**_tempoFilterTypeFib_** : filtrage temporel en entrées (pour images fixes) moyenne, moyenne coupée, médiane, max, min

**_calibration_** : calibration des taux d'injection

**_outputSigmaFilter_ :** filtrage passe bas en sortie.

Ces paramètres doivent apparaître clairement. Les autres doivent juste être accessibles, au cas où et par soucis d'exhaustivité.

## Références

[1] Aymeric Perchant. Manuel de référence pour imagecell. Technical report, Mauna Kea Technologies, Base CVS, SoftTex/UtilisationSoutien/ManuelReference/ReferenceImageCell.tex, 2002.

**Diligence Document 10**

Destinataires : Dpt. Image — MAUNA KEA TECHNOLOGIES

Auteurs : Aymeric Perchant, Georges Le Gonalher

Sujet : **Traitement de l'Image** — 4 septembre 2001

## introduction

Les images acquises par le tomoscope sont bruitées. La principale composante de ce bruit semble être le bruit poissonnien issu du photodétecteur. D'autres source de bruit sont envisagés comme le speckle, ou bien le bruit d'autres appareils (électronique, optiques, quantification, contrôle du gain...). Ce document a pour but de trouver la meilleur opération de type moyennage permettant à partir de plusienrs aquisistions fixes (pas de bouger) de diminuer la variance du signal. Nous ne nous attacherons pour le moment qu'à étudier des opération essentiellement temporelles.

Le principe de reconstruction de l'image à partir du jeu de mesures fait intervenir plusieurs spots dans un pixel. Le problème ici est de savoir comment calibrer et combiner ces différents spots pour pouvoir obtenir le pixel le moins bruité. Le deuxième problème est d'optimiser le moyennage temporel de plusieurs jeux de mesures. Ces deux problèmes sont évidemment liés.

# 1 Cas Simple

On veut calculer une calibration en intensité et un mapping géométrique pour corriger respectivement la modulation du signal par le guide d'image et la distorsion géométrique induite par la vitesse sinusoïdale du miroir ligne.
Le cas le plus simple consiste à considérer que l'on a pas de bruit sur la mesure au niveau du PM, que le balayage est reproductible et que l'on peut calculer la position du spot dans le repère de balayage. C'est cette méthode qui est implémentée dans la version actuelle du Simulateur SIMUAPP (cf. `CalibrationMapping` et `ReconstructionMapping`).

On considère :
- $cal(k)$ : le $k^{ieme}$ élément du flot de données obtenu en observant un miroir. Il s'agit d'un ensemble de 512X512 éléments par exemple.
- $obs(k)$ : le $k^{ieme}$ élément du flot de données obtenu en observant l'échantillon d'intérêt.
- $cal^j(k)$ (ou $obs^j(k)$) : le $k^{ieme}$ élément du flot de l'acquisition j. L'indice j représente donc un indice temporel.

Initialiser les éléments de 3 vecteurs :

```
pour tout k
  X[k] = Y[k] = coeff[k] = INVALIDE

pour tout k
 si cal[k] != 0
  calculer l'indice du pixel (X[k], Y[k]) de l'image reconstruite dans lequel tombe ce spot.
    si cet indice est valide :
       coeff[k]  = (1/cal[k]) si on veut faire une calibration en amplitude
       coeff[k]  = 1.0 sinon
       pond(X[k], Y[k]) +=1;
```

pond() est une image de même taille que l'image reconstruite qui indique le nombre de spot valide qui tombe dans chacun des pixels.

Pour tout élément du flot calculer :

```
coeff[k]  *= 1.0/pond(X[k], Y[k])
```

La calibration calcule donc 3 vecteurs : coeff[k] , X[k] , Y[k].

Reconstruction :

Pour tout élément du flot obs(k) :

```
si X[k], Y[k] sont valides:
  si coeff[k] est valide
     image(X[k], Y[k] ) += obs[k]*coeff[k];
```

Dans le cas où l'on dispose de N images d'acquisitions et de K images pour calculer la calibration. On remplace les flots obs(k) et cal(k) par leur moyenne temporelle :

$$obs(k) = \frac{1}{N}\sum_j obs^j(k)$$

$$cal(k) = \frac{1}{K}\sum_j cal^j(k)$$

# 2 Modèle

## 2.1 Distribution des amplitudes dues au guide d'image

Dans cette étude, le terme amplitude désigne la valeur mesurée du signal en sortie du tomoscope. Ce qui nous intéresse est dans le modèle de bruit est le rapport entre l'amplitude reçue si on avait ni guide d'image, ni trou de filtrage, et l'amplitude théoriquement reçue en présence de ces deux composantes du système.
Nous avons d'abord étudié la distribution des amplitudes maximales de l'échantillonnage du guide d'images. La trajectoire du laser passe par l'entrée du guide d'image, ce qui module l'amplitude de la puissance du spot en fonction de l'intersection des disques (spot et fibre optique). Puis, l'amplitude est encore modifiée au retour par le trou de filtrage en fonction encore une fois de l'intersection de disques.
Le balayage laser est supposé à puissance (pulsée) identiques partout. Le système de balayage va créer une certaine distribution d'amplitudes dues aux deux atténuations, dont la distribution de probabilité d'amplitude est donnée sur la figure 1. Nous noterons $p_a(x)$ cette distribution discrète.

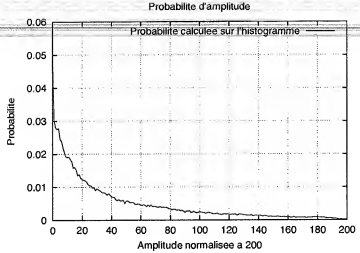



FIG. 1 – Distribution de probabilité des amplitudes due au phénomène d'acquisition via l'échantillonnage du guide d'image, puis le trou de filtrage, avec les paramètres par défaut du tomoscope. Cette distribution est estimée à partir de l'histogramme des mesures simulées.

Cette distribution a été calculée à partir de donnée non bruitées issues du simulateur, pour les paramètres par défaut de celui-ci, ainsi que pour une normalisation de l'amplitude de sortie à 200. La figure 2 représente la fonction de répartition de la distribution de probabilité $p_a$.. C'est-à-dire pour une amplitude donnée $X_0$, on représente $P(x < X_0)$.
Dans les expériences qui suivent, nous utiliserons cette probabilité $p_a$ afin de simuler plusieurs amplitudes *théoriques*, c'est-à-dire on bruitées, de plusieurs spots. Le principe de reconstruction de l'image à partir du jeu de mesures fait intervenir plusieurs spots dans un pixel. Les amplitudes théoriques de ces spots seront calculées avec la distribution $p_a$.

## 2.2 Rappels sur le bruit poissonnien du système

Nous tenons compte ici des bilans photométriques expliqués dans le document du simulateur (Georges Le Goualher, Grégoire Malandain, Aymeric Perchant), ainsi que dans le bilan de fin d'année de Magalie Surivet. Les notions sont les mêmes qu pour le document du simulateur. Ce bilan peut se résumer ainsi.

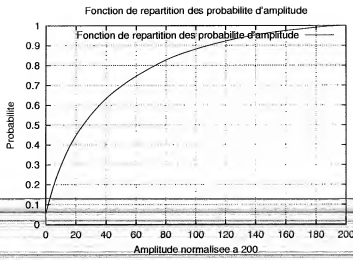



FIG. 2 – Fonction de répartition des amplitudes. On remarque que 50% des amplitudes sont inférieures à 25, soit 12.5% de l'amplitude maximum.

| Bilan photométrique | Paramètres | | |
|---|---|---|---|
| | Id | Défaut | Domaine |
| **Puissance dans le guide d'image** | $P_{guide}$ | $770\mu W$ | $[0.01, 5]mW$ |
| **Rapport cyclique** | $\frac{T_{pulse}}{T_{emission}}$ | 4 | |
| **Coefficient de réflexion parasite** | $r_p$ | $0.55\ 10^{-4}$ | |
| **Coefficient de rétrodiffusion maximale** | $R_{max}$ | 0.001 | |
| **Coefficient de réflexion au point considéré** | $R(x)$ | $R_{air/tissu}$ | $[0, 017; 0, 04]$ |
| **Coefficient d'atténuation de la rétrodiffusion** | $a_r$ | 0.103447 | |
| **Energie d'un photon** | $E1\lambda(\lambda)$ | $2.89\ 10^{-19}$ | |

Le nombre de photons qui vont être compté par le photomultiplicateur est au total de :

$$N(R(x)) = N_{r_p} + N_{utile} = N_{guide}\Big(r_p + a_r * R_{max} * R(x)\Big),$$

avec

$$\begin{aligned} N_{r_p} &= N_{guide} * r_p, \\ N_{utile} &= N_{guide} * a_r * R_{max} * R(x), \\ N_{guide} &= \frac{P_{guide} T_{emission}}{E_{1\lambda}} \frac{T_{pulse}}{T_{emission}}. \end{aligned}$$

Le signal $N(R(x))$ est alors entaché d'un bruit poissonnien. Le signal bruité est noté $\tilde{N}(R(x))$. Ce signal comporte un offset par rapport à la partie utile que nous retirons, puis nous appliquons une correction du gain avec :

$$\hat{R}(x) = \frac{\tilde{N}(R(x)) - N_{guide} * r_p + k * \sqrt{\tilde{N}}}{a_r * R_{max}}. \quad (1)$$

afin de retrouver un estimé noté $\hat{R}(x)$ de $R(x)$. Le facteur entier $k$ règle un offset supplémentaire exprimé en terme d'écart-type du bruit. Ceci permet de ne pas saturer le signal si le bruit est important.

## 2.3 Mesures bruitées

Un pixel est reconsitué à partir en moyenne de 8 mesures de spots pour les paramètres par défaut. Nous noterons $N_{spot/pixel}$ ce nombre. Chacun de ces spots a une amplitude modulée par le système guide d'image + trou de filtrage. Nous utiliserons la distribution de probabilité $p_a$ pour simuler des répartitions.
Nous notons $\left(a_i\right) = \left\{a_i,\ i \in \{1..N_{spot/pixel}\}\right\}$ l'ensemble des amplitudes (non bruitées) des $N_{spot/pixel}$ spot du pixel considéré. Ces amplitudes sont uniquement dues aux passages successifs de la lumière dans le guide d'image, puis dans le trou de filtrage. De plus celles-ci sont par défaut des coefficient d'atténuation entre dans $[0; 1]$, ou bien lorsque cela est précisé, les $a_i$ sont normalisés dans $[0; 200]$ afin de respecter la dynamique de 8 bits du système (cf. figures).

### 2.3.1 Mesures sur tissu

Pour un tissu, nous supposons que nous avons $R_{max} \approx 0.001$. Les variations de la qualité de couplage, modélisées par les coefficients $a_i$, influent à la fois sur la réflection parasite et sur le signal utile. En tenant compte de la variation d'amplitude $a_i$ du système, les équations précédentes deviennent :

$$N_i(R(x)) = a_i * N_{guide}\Big(r_p + a_r * R_{max} * R(x)\Big),$$

avec l'indice i qui représente le ième spot qui intersecte dans le pixel. Cette amplitude théorique est perturbée par le bruit de poisson pour produire une variable aléatoire poissonnienne $\tilde{N}_i(R(x))$. $N$ étant grand, on peut approcher cette variable par la loi normale :

$$\tilde{N}_i(R(x)) \rightsquigarrow \mathcal{N}\Big(a_i * N_{guide}(r_p + a_r * R_{max} * R(x)), \sqrt{a_i * N_{guide}(r_p + a_r * R_{max} * R(x))}\Big).$$

Nous noterons avec l'exposant $(j)$ les réalisations de cette variable $\tilde{N}_i^{(j)}(R(x))$, $j \in \{1, ..., N_N\}$. Par comodité ces réalisations sont notées simplement $\tilde{N}_i^{(j)}$ lorsqu'il n'y a pas d'ambiguité.
Ces réalisations sont ensuite normalisés linéairement et quantifiées sur 8 bits. En pratique les $\tilde{N}_i^{(j)}$ varient entre 0 et $N_{guide} * (r_p + a_r * R_{max})$. La normalisation doit prendre en compte deux phénomènes. D'abord, la distribution des coefficient $a_i$ module directement l'amplitude des $\tilde{N}_i^{(j)}$ : cela implique que toute la dynamique de ceux-ci est une dynamique utile. Ensuite, pour un $a_i$ fixé la dynamique réelle à utiliser n'est pas $[0; a_i * N_{guide}(r_p + a_r * R_{max})]$ mais $[a_i * N_{guide}r_p; a_i * N_{guide}(r_p + a_r * R_{max})]$.
On ne peut cependant pas connaître a priori quel est la valeur du coefficient de couplage $a_i$ pour une mesure $\tilde{N}_i^{(j)}$, et il faut donc ne prendre en compte que le premier phénomène pour la correction de la dynamique, et donc y intégrer la composante parasite qui sera retirée avec les procédures de calibration en amplitude ultérieures.
Plusieurs méthodes sont possibles :
- méthode des quartiles (1er quart jusqu'à 0.04, 2nd quart 0.125 , 3ème quart 0.305),
- normalisation max/min,

$$\max_{R(x),a_i}(\tilde{N}_i^{(j)}) = N_{guide} * (r_p + a_r * R_{max})$$

$$\min_{R(x),a_i}(\tilde{N}_i^{(j)}) = 0$$

- normalisation moyenne (ou médiane) : point fixe. Moyenne $= 0.2$, médiane $= 0.125$, point fixe=???.

Idée du moment... : prendre un certain pourcentage à partir de 0 car les signaux les plus faibles sont aussi les plus nombreux et donc les plus importants pour une reconstruction des pixels. Les trois premiers quartiles représentent 30.5% de la dynamique maximale si on ne prend pas en compte le coefficent de rétrodiffusion $R(x)$, soit en nombre de photons : $0.305 * N_{guide} * (r_p + a_r * R_{max}) = 15461$. Si on prend 90% de la dynamique, on a alors $0.54 * N_{guide} * (r_p + a_r * R_{max}) = 27373$ photons à compter.
Voici un tableau de correspondance entre le pourcentage de la dynamique et le pourcentage de mesure que cela représente :

| % de mesures | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 95 | 98 | 99 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| % de la dynamique | 1 | 3 | 5.5 | 8.5 | 12.5 | 18 | 25.5 | 36.5 | 54 | 69 | 83.5 | 90 | 100 |

En saturant à 54% de la dynamique les amplitudes (très improbables) les plus grandes (donc on en garde 90% bien calibrées) la normalisation devient :

$$\tilde{n}_i^{(j)}(R(x)) = \frac{\tilde{N}_i^{(j)}}{0.54 * N_{guide} * (r_p + a_r * R_{max})}.$$

avec les $\tilde{n}_i^{(j)}(R(x)) \in [0;1]$, les valeurs dépassant les bornes étant alors saturées sur ces bornes.
Avec cette normalisation, les $\tilde{n}_i^{(j)}(R(x))$ suivent la loi normale :

$$\tilde{n}_i^{(j)}(R(x)) \rightsquigarrow \mathcal{N}\Big(\frac{a_i * (r_p + a_r * R_{max} * R(x))}{0.54 * (r_p + a_r * R_{max})}, \frac{\sqrt{a_i * N_{guide}(r_p + a_r * R_{max} * R(x))}}{0.54 * N_{guide} * (r_p + a_r * R_{max})}\Big).$$

Ainsi, en saturant à la moitié de la dynamique maximum, on diminue et on homogénéise le bruit.

### 2.3.2 Mesure sur miroir (calibration)

Nous pouvons réaliser les mêmes mesures sur un miroir ; $R_{max} = 1$ et $R(x) = 1$. Les notations sont similaires ; on remplace la lette $N$ par la lettre $C$. En tenant compte de la variation d'amplitude $a_i$ du système :

$$C_i(R(x)) = a_i * N_{guide}\Big(r_p + a_r\Big),$$

avec l'indice i qui représente le ième spot qui intersecte dans le pixel. Cette amplitude théorique est perturbée par le bruit de poisson pour produire une variable aléatoire poissonnienne $\tilde{C}_i$. $C$ étant grand, on peut approcher cette variable par la loi normale

$$\tilde{C}_i \rightsquigarrow \mathcal{N}\Big(ai * N_{guide}(r_p + a_r), \sqrt{a_i * N_{guide}(r_p + a_r)}\Big).$$

Nous noterons avec l'exposant $(j)$ les réalisations de cette variable $\tilde{C}_i^{(j)}$, $j \in \{1, ..., N_C\}$. Ces réalisations sont normalisés avec :

$$\tilde{c}_i^{(j)} = \frac{\tilde{C}_i^{(j)}}{N_{guide}(r_p + a_r)}.$$

Nous avons donc $\tilde{c}_i^{(j)}$ qui est une réalisation d'une variable aléatoire

$$\tilde{c}_i^{(j)} \rightsquigarrow \mathcal{N}\Big(a_i, \sqrt{\frac{a_i}{N_{guide}(r_p + a_r)}}\Big).$$

### 2.3.3 Problème 1 du moyennage

Le problème peut alors se formuler ainsi :

> **Etant donnés un premier jeu de mesures acquises sur un miroir $\left(\tilde{c}_i^{(j)}\right)$ et un second jeu de mesures acquises sur un tissu $\left(\tilde{n}_i^{(j)}(R(x))\right)$, quel est le meilleur estimateur $\hat{R}(x)$de $R(x)$ ?**

On peut a priori distinguer trois méthodes génériques :

**Moyennage temporel, puis sur le pixel :** la calibration $\left(\tilde{c}_i^{(j)}\right)$ doit permettre d'estimer chaque $a_i$. Si on connait $a_i$, alors on peut estimer plus facilement $R(x)$ à partir de $\left(\tilde{n}_i^{(j)}(R(x))\right)$.

**Moyennage sur le pixel puis temporel :** le pixel est reconstruit pour les données de calibration, et pour les données sur tissu, et ce à chaque instant de mesure. Puis la succession de pixels dans le temps est moyenné pour atténuer le bruit.

**Moyennage temporel et sur le pixel simultanément :** le moyennage utilisé s'effectue simultannément sur le pixel et dans le temps.

Un point commun des méthode est que l'on cherche à s'afranchir des $a_i$ dans l'estimation des $\bar{N}_i$ afin de pouvoir retrouver $\hat{R}(x)$. On cherche donc à estimer :

$$\tilde{R}(x) = \frac{\bar{n}}{\bar{c}} = \frac{r_p + a_r * R_{max} * R(x)}{r_p + a_r * R_{max}},$$

où $\hat{R}(x)$ est un estimateur biaisé de $R(x)$. La calibration ne doit cependant pas s'arrêter là car il reste un offset à soustraire, puis il faut rétablir la dynamique. On réalise donc l'opération suivante pour ôter le biais de $\hat{R}(x)$ :

$$\hat{R}(x) = \left(\tilde{R}(x) - \frac{r_p}{r_p + a_r * R_{max}}\right) * \left(1 + \frac{r_p}{a_r * R_{max}}\right).$$

## 3 Moyennage temporel, puis sur le pixel

$$\hat{R}_i(x) = \frac{N_G}{N_N} \frac{\sum_{j=1}^{N_N} \bar{n}_i^{(j)}(R(x))}{\sum_{j=1}^{N_G} \bar{c}_i^{(j)}}. \tag{2}$$

La somme du numérateur correspond à l'estimateur du maximum de vraisemblance qui permet d'estimer le coefficient $a_i$. On divise ensuite chaque terme de la somme du numérateur par l'estimé de $a_i$. La somme du numérateur est alors elle aussi un estimateur du maximum de vraisemblance de $R(x)\frac{a_r}{\hat{a}_i}$.
Une fois que l'on a un estimé de la valeur réelle de chaque spot, il faut les intégrer afin de créer un pixel de l'image reconstruite. Nous proposons plusieurs méthodes simples :

$$\tilde{R}(x) = \frac{1}{N_{spot/pixel}} \sum_i \hat{R}_i(x). \tag{3}$$

$$\check{R}(x) = \text{Median}_i\{\hat{R}_i(x)\} \tag{4}$$

$$\hat{R}(x) = \text{OWA}_i\{\hat{R}_i(x)\} \tag{5}$$

Avec $\text{OWA}_i$ l'opérateur de moyenne pondérée ordonnée. Soient $(x_i)$ une suite de $N$ valeurs réelles et $(y_i)$ la suite des valeurs $(x_i)$ mais ordonnée par ordre selon une relation d'ordre prédéfinie. Alors :

$$\text{OWA}_i(x_i) = \frac{2}{N(N+1)} \sum_{i=1}^{N} i.x_i.$$

L'OWA permetde privilégier les valeurs qui sont plus grandes que les autres par rapport à la relation d'ordre. Dans notre cas précis la relation d'ordre est équivalente à la relation d'ordre "$<$" des amplitudes estimées $\hat{a}_i$ à partir des $\bar{c}_i^{(j)}$.

## 4 Moyennage de spots pour reconstruire un pixel

$$\hat{R}^{(j)}(x) = \frac{\sum_{i=1}^{N_{spot/pixel}} \bar{n}_i^{(j)}}{\frac{1}{N_G}\sum_{j=1}^{N_G}\sum_{i=1}^{N_{spot/pixel}} \bar{c}_i^{(j)}}. \tag{6}$$

Il faut remarquer ici qu'en l'absence du bruit de poisson, et donc en remplaçant les variables $\bar{n}_i^{(j)}$ et $\bar{c}_i^{(j)}$ par leurs moyennes respoectives $a_i R((x))$ et $a_i$ l'opération serait :

$$\tilde{R}^{(j)}(\boldsymbol{x}) = \frac{\sum_{i=1}^{N_{spot/pixel}} a_i R((\boldsymbol{x}))}{\frac{1}{N_C}\sum_{j=1}^{N_C}\sum_{i=1}^{N_{spot/pixel}} a_i} = \frac{\sum_{i=1}^{N_{spot/pixel}} a_i R((\boldsymbol{x}))}{\sum_{i=1}^{N_{spot/pixel}} a_i}.$$

Cet opérateur est un opérateur de moyenne pondérée en fonction de l'importance des $a_i$. L'équation 6 représente un opérateur de moyenne pondéré par l'estimé $\hat{a}_i$. Cette première opération permet de tenir directement compte de la fiabilité de chaque mesure, cette fiabilité étant exprimée par l'amplitude $\hat{a}_i$. Cet amplitude règle le niveau du signal reçu, et l'importance du bruit poissonnien associé. L'équation 6 est une moyenne pondérée où les mesures les moins bruitées ont un poids plus important que les mesures bruitées. Nous disposons donc de mesures du pixel à chaque instant d'acquisition. Il reste à les moyenner temporellement afin de réduire le reste du bruit. Nous proposons alors les opérateurs vu précédement :

$$\tilde{R}(\boldsymbol{x}) = \frac{1}{N_N}\sum_{j} \tilde{R}_{(j)}(\boldsymbol{x}). \tag{7}$$

$$\tilde{R}(\boldsymbol{x}) = \text{Median}_j\{\tilde{R}_{(j)}(\boldsymbol{x})\} \tag{8}$$

$$\tilde{R}(\boldsymbol{x}) = \text{OWA}_j\{\tilde{R}_{(j)}(\boldsymbol{x})\} \tag{9}$$

# 5 Quantification

Tous ces calculs sont à vérifier par un expert du signal ?

## 5.1 Rapport signal sur bruit de la quantification

On peut modéliser la distribution des amplitudes des $a_i$ par une loi exponentielle décroissante. On a alors :

$$p_{a_i}(x) = \alpha e^{-\alpha x},$$

avec $\alpha \approx \frac{4.002}{N_{max}}$, et $N_{max}$ est le niveau maximum de la quantification (255 en 8 bits), le niveau min étant 0 par défaut. On divise l'intervalle $[0; N_{max}]$ en $2^N$ intervalles de longueurs identiques $q = \frac{N_{max}}{2^N}$. Le rapport signal sur bruit est donné en dB par :

$$SNR_{dB} = 10\log_{10}\left(\frac{P_s}{P_b}\right),$$

avec $P_s$ la puissance du signal et $P_b$ la puissance du bruit. On a alors pour le bruit de quantification :

$$P_b = \frac{q^2}{12} = \frac{N_{max}^2}{12\ 2^{2N}}$$

et pour signal :

$$\begin{aligned} P_s &= \left(\int_0^\infty x^2 \alpha e^{-\alpha x} dx\right) - \left(\int_0^\infty x \alpha e^{-\alpha x} dx\right)^2 \\ &= \frac{2}{\alpha^2} - \left(\frac{1}{\alpha}\right)^2 \\ &= \frac{1}{\alpha^2} \\ &\approx \frac{N_{max}^2}{16}. \end{aligned}$$

Le rapport signal sur bruit vaut donc :

$$SNR_{dB} = 10log_{10}\frac{3}{4} + 10log_{10}2^{2N} = 6N - 1.25$$

Soit $46.75dB$ sur 8 bits.

### 5.2 Rapport signal sur bruit du au bruit de poisson

Si on suppose que le bruit de poisson est le bruit prédominant, la puissance du signal non bruité vaut $\lambda^2$, $\lambda$ étant la valeur exacte du signal et donc le paramètre du bruit de poisson. La puissance du bruit de poisson vaut $\lambda$ (moment centré d'ordre 2). Le rapport signal sur bruit s'obtient donc facilement avec :

$$SNR_{dB} = 10log_{10}\lambda.$$

Le bruit de poisson est un bruit de comptage du signal, et $\lambda$ représente bien ici le nombre de photon comptés dans le photomultiplicateur (PM). On ne prend pas en compte ici uniquement le signal utile mais le signal entier car on ne peut pas différencier les deux avant l'étape de calibration informatique. Il faut donc tout quantifier. Avec les paramètres par défaut on compte environ 56000 photons dans le PM, et donc $SNR_{dB} = 47dB$.

### 5.3 Bilan

On constate facilement que le niveau du bruit de poisson et le niveau de la quantification sont équivalents. De deux choses l'une : soit on croit au père noël et on suppose que la calibration sera parfaite et que l'on aura pas de perte, soit on rajoute quelques bits pour être confortable...
Dans le premier cas (on a cru au père noël mail il n'est pas venu) il vaut mieux saturer les niveaux hauts et donc bien quantifier les niveaux bas car ils sont plus nombreux. Dans le second cas, la question est : combien faut-il rajouter de bits pour être confortable ? Ma réponse intuitive est qu'il faut 10 ou 12 bits au total.
Dernière question : quel gain peut-on espérer par bit rajouté ?

## 6 Résultats

Les principaux résultats attendus sont :
- comportement de la quantification sur 8 bits en fonction du contrôle du gain,
- forme et comportement de la calibration (ce point est étroitement lié à la forme du moyennage),
- comportement des moyennages en fonction de :
  - $N_N$ et $C_N$,
  - la fonction de moyennage (moyennage temporel d'abord, ou pixel d'abord),
  - du nombre de spots dans un pixel,
  - du coefficient de rétrodiffusion observé.

Le nombre de paramètres est assez important, mais nous pouvons tout de même dégager un certain nombre de résultats.

### 6.1 Comparaison des deux types de méthodes de moyennage

On observe $R(x) = 0.5$. Les paramètres utilisés dans ce paragraphe :

| | |
|---|---|
| $R(x) = 0.5$ | → c'est le coefficient *normalisé* dans $[0, 1]$, soit une valeur réelle de 128 ici. |
| $N_C = 5$ | → nombre d'acquisitions pour la calibration |
| $N_N = 5$ | → nombre d'acquisitions pour la mesure sur tissu |
| $N_{spots/pixel} = 3$ ou 8 | → nombre moyen de spots pour un pixel dans le cas réel |
| Statistiques sur 1000 mesures | → les statistiques sont significatives |

La calibration utilisée est la calibration optimale par défaut..
On simule la reconstruction du pixel avec les deux types de méthodes. On répete l'expérience 1000 fois afin de faire des statistiques. Les histogrammes des résultats obtenus sont représentés sur les figures 3 et 4. Le tableau suivant donne les moyennes, les medianes et l'écart typde des mesures obtenues pour chaque méthode ($N_{spots/pixel} = 8$).

| Méthode | Moyenne | Médiane | Ecart-type |
|---|---|---|---|
| Moy. temp. puis moy. pixel | 130.5 | 128 | 7.2 |
| Moy. temp. puis median pixel | 127.4 | 127.5 | 1.4 |
| Moy. temp. puis OWA pixel | 128.2 | 127.7 | 2.2 |
| Moy. pixel puis moy. temp. | 127.6 | 127.6 | 0.9 |
| Moy. pixel puis median temp. | 127.6 | 128.0 | 1.0 |
| Moy. pixel puis OWA temp. | 127.8 | 127.9 | 0.9 |

On remarque que les moyennes temporelles sont bien moins performantes que l'intégration dans un pixel. Cela peut s'expliquer par le fait que l'intégration dans un pixel est implicitement réalisé par une moyenne pondérée par l'importance intrinsèque du bruit poissonien de la mesure. Le moyennage temporel s'effectue lui sur des données forcément mal calibrées et se révèle moins bon.
Le tableau suivant donne les moyennes, les medianes et l'écart typde des mesures obtenues pour chaqne méthode avec $N_{spots/pixel} = 3$ (les figures correspondantes sont les figures 5 et 6).

| Méthode | Moyenne | Médiane | Ecart-type |
|---|---|---|---|
| Moy. temp. puis moy. pixel | 130.9 | 127.7 | 12.7 |
| Moy. temp. puis median pixel | 128.2 | 128.0 | 8.1 |
| Moy. temp. puis OWA pixel | 129.3 | 127.7 | 7.2 |
| Moy. pixel puis moy. temp. | 127.8 | 127.6 | 2.5 |
| Moy. pixel puis median temp. | 127.8 | 128.0 | 3.3 |
| Moy. pixel puis OWA temp. | 128.3 | 128.0 | 2.7 |

On remarque ici que les résultat enter les deux familles de méthodes sont plus proches. L'intégration dans le pixel semble encore meilleure, au moins pour la moyenne. Ce test ne change pas a conclusion, mais montre bien la sensibilité de la reconstruction par rapport au facteur de sur-échantillonnage.

### 6.2 Estimation des paramètre "optimaux"

A FAIRE :
- déterminer le nombre d'images nécessaires à la calibration,
- déterminer le nomnre d'image nécessaire pour les mesures sur tissu,
- quantifier l'influence du décalage lors de la calibration,
- étude théorique analytique de la calibration,
- influence de la répartition des amplitudes des spots,
- ...

## 7 Conclusion sur le moyennage

La première conclusion qui s'impose est le type de moyennage a effectuer : les données de calibrations doivent être globalement moyennées avec :

$$\frac{1}{N_G}\sum_{j=1}^{N_G}\sum_{i=1}^{N_{spot/pixel}} \hat{c}_i^{(j)},$$

puis, utiliser cette "calibration" pour intégrer les mesures de chaque acquisition dans le pixel. Le moyennage temporel est effectué en dernier en moyennant les pixels des images consécutives. Ce dernier type de moyennage importe peu car les résultats obtenus dans la partie 6.1 son similaires pour la moyenne arithmétique, la médianne, et l'OWA. Le plus simple est donc d'utiliser la moyenne arithmétique qui représente le coût de calcul le plus faible.

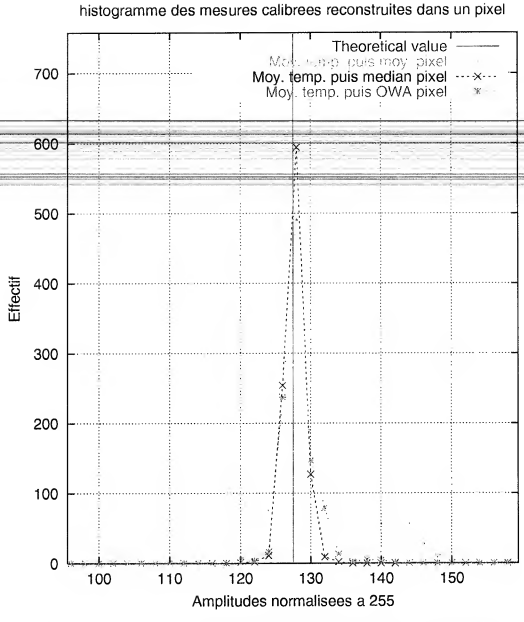



FIG. 3 – Histogramme des mesures calibrées et reconstruites dans un pixel ($N_{spots/pixel} = 8$) pour les paramètres de la partie 6.1. On effectue d'abord le moyennage temporel, puis l'intégration dans le pixel avec les trois méthodes différentes.

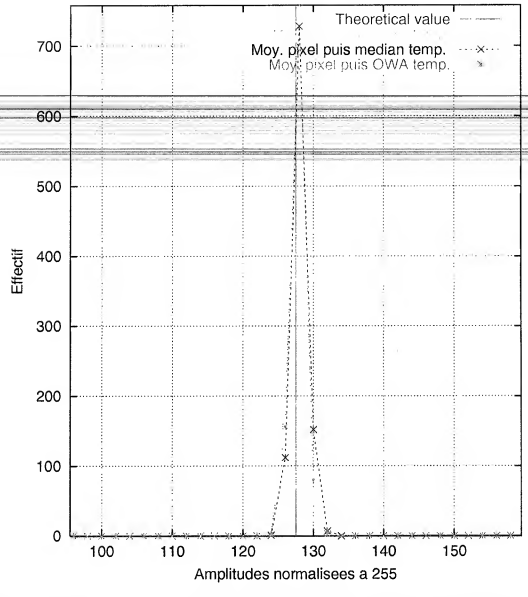



FIG. 4 – Histogramme des mesures calibrées et reconstruites dans un pixel ($N_{spots/pixel} = 8$) pour les paramètres de la partie 6.1. On effectue d'abord l'intégration dans le pixel, puis le moyennage temporel du pixel avec les trois méthodes différentes.

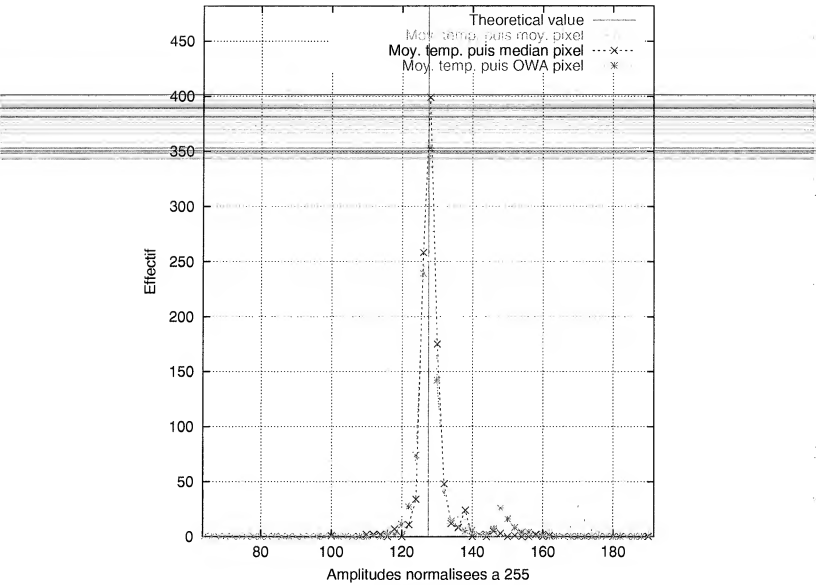



FIG. 5 – Histogramme des mesures calibrées et reconstruites dans un pixel pour les paramètres de la partie 6.1 ($N_{spots/pixel} = 3$). On effectue d'abord le moyennage temporel, puis l'intégration dans le pixel avec les trois méthodes différentes.

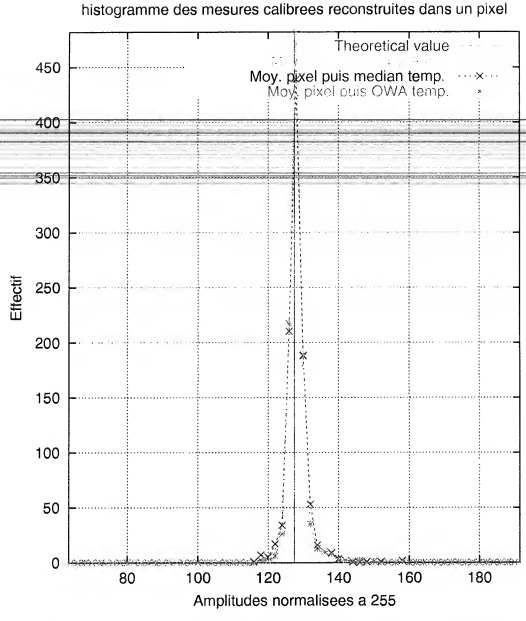



FIG. 6 – Histogramme des mesures calibrées et reconstruites dans un pixel pour les paramètres de la partie 6.1 ($N_{spots/pixel} = 3$). On effectue d'abord l'intégration dans le pixel, puis le moyennage temporel du pixel avec les trois méthodes différentes.

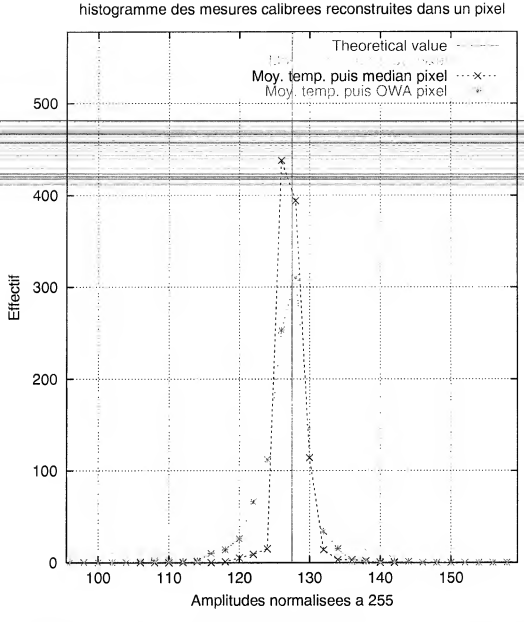



FIG. 7 – Histogramme des mesures calibrées et reconstruites dans un pixel pour les paramètres de la partie 6.1 ($N_{spots/pixel} = 8$). On effectue d'abord le moyennage temporel, puis l'intégration dans le pixel avec les trois méthodes différentes. Les mesures sont sur-normalisées (=trop faibles).

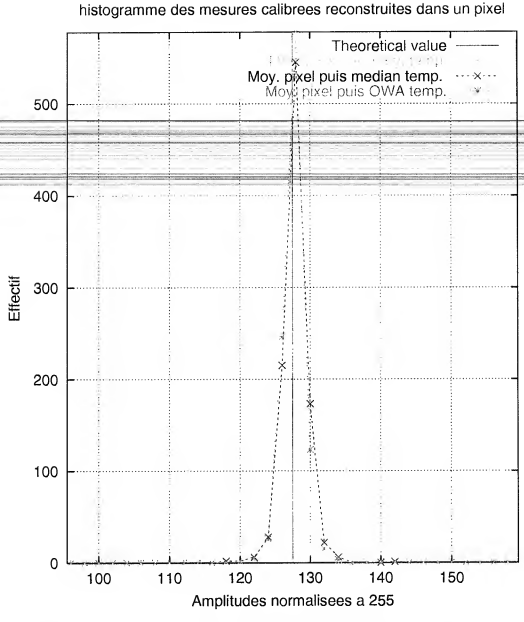



FIG. 8 – Histogramme des mesures calibrées et reconstruites dans un pixel pour les paramètres de la partie 6.1 ($N_{spots/pixel} = 8$). On effectue d'abord l'intégration dans le pixel, puis le moyennage temporel du pixel avec les trois méthodes différentes. Les mesures sont sur-normalisées (=trop faibles).

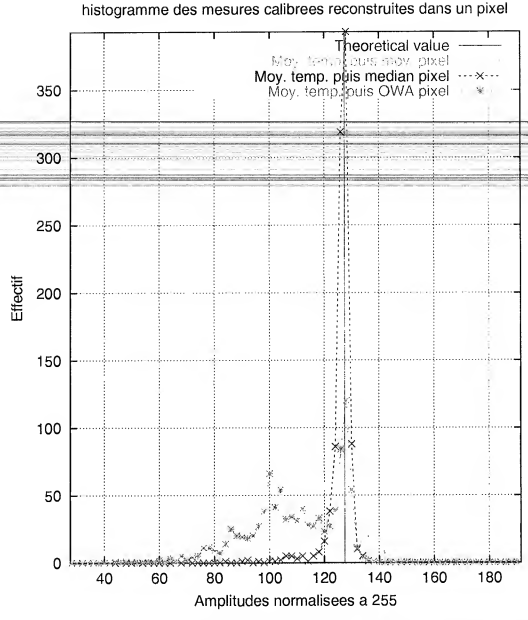



FIG. 9 – Histogramme des mesures calibrées et reconstruites dans un pixel pour les paramètres de la partie 6.1 ($N_{spots/pixel} = 8$). On effectue d'abord le moyennage temporel, puis l'intégration dans le pixel avec les trois méthodes différentes. Les mesures sont sous-normalisées (=saturées).

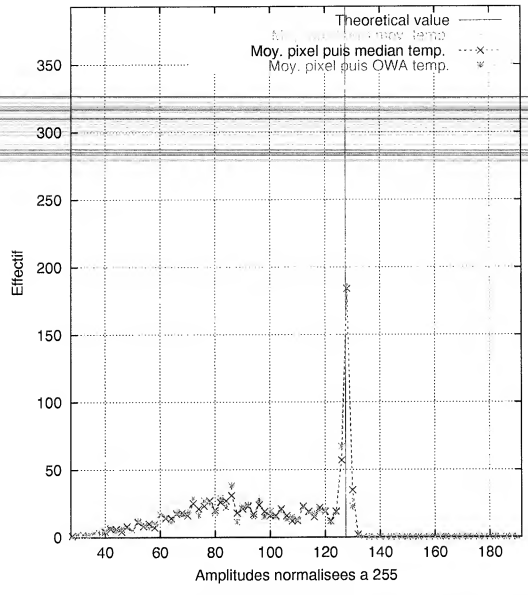



FIG. 10 – Histogramme des mesures calibrées et reconstruites dans un pixel pour les paramètres de la partie 6.1 ($N_{spots/pixel} = 8$). On effectue d'abord l'intégration dans le pixel, puis le moyennage temporel du pixel avec les trois méthodes différentes. Les mesures sont sous-normalisées (=saturées).

# 8 Restauration sans calibration

Cette partie propose une méthode de restauration des images de mesures n'utilisant pas le calibrage en amplitude. Le but est de s'affranchir des phénomène d'échantillonnage sur le guide d'images. Les deux principaux phénomènes sont :

**le problème d'injection non uniforme :**

**aliasing :** du à une fréquence d'échantillonnage inférieure à la fréquence de Nyquist.

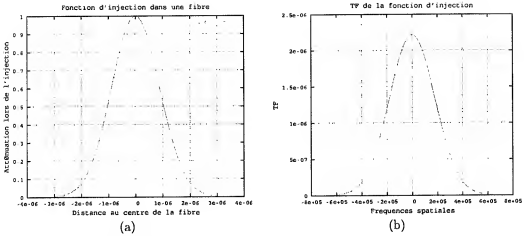
## 8.1 Analyse dans le plan de fourrier

Supposons que le spot balaye continument une seule fibre, sur toute son diamètre. Alors la fonction d'injection est (pour un spot du même diamètre que la fibre) :

$$a_i(x) = e^{-\frac{x^2}{r_{fibre}^2}},$$

en supposant la fibre centrée en $x = 0$, alors x indique la position du spot par rapport à la fibre, et $r_{fibre}$ est le rayon de la fibre. Cette courbe est représentée sur la figure 11-(a). La figure 11-(b) représente alors sa TF qui est aussi une gaussienne :

$$TF(a_i)(\nu) = r_{fibre}\ \sqrt{\pi}\ e^{-\pi^2\ \nu^2\ r_{fibre}^2}.$$

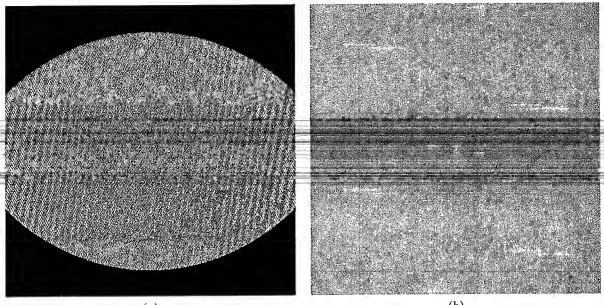



(a) (b)

FIG. 11 – (a) est la fonction d'injection dans une fibre en fonction de la distance entre le centre du spot laser et le centre de la fibre. (b) est la TF de (a).

Soit $\nu_c$ la fréquence de coupure approchée. La portion de gaussiène repliée (aliasing) a la forme d'une queue de gaussienne $erfc(\frac{1}{\sqrt{2}}\frac{\nu_c}{\sigma_\nu})$. On peut donc approcher une fréquence de coupure à $\nu_c = 3\sigma_\nu = \frac{3}{\sqrt{2}\pi r_{fibre}} = 5.40\ 10^5$. On en déduit la fréquence d'echantillonnage optimale (spatiale) à $\frac{6}{\sqrt{2}\pi r_{fibre}}$, ce qui donne une période spatiale «optimale» de $0.93\mu m$ pour $r_{fibre} = 1.25\mu m$. Si on prend $\nu_c = 2\sigma_\nu$, alors la période spatiale est de $1.38\mu m$.
La fibre étant circulaire, on peut supposer que cette fréquence spatiale doit être respectée en $x$ et en $y$.
En pratique la période temporelle d'échantillonnage est $120ns$. Si on suppose une vitesse constante du spot d'environ $11ms^{-1}$, cela donne une période spatiale de $1.32\mu m$ en $x$. En $y$ la période spatiale est variable en fonction de la vitesse du miroir galva. Les paramètres par défaut donnent une fréquence de raffraichissement de 15 images par seconde, avec une période ligne de $125\mu s$. En supposant un balyage sur $800\mu m$ la période spatiale en $y$ est de $1.5\mu m$.

Ces chiffres indiquent que l'on va faire apparaître de l'alisasing sur les images, avec une fréquence en $x$ différente de la fréquence en $y$. L'aliasing sera plus important suivant la direction y.
La figure 12-(a) représente la TF de l'image 12(b) qui est une image simulée, avec le guide d'image tourné de 10 degrés, qui image une surface uniforme, et avec un échantillonnage instable à $1.5\mu m$ près pour chaque mesure.



(a) (b)

FIG. 12 – (a) Image. (b) FFT de (a).

La forme centrale est due à la forme du guide. Les 6 pics les plus importants sont le signal utile mélangé aux pics représentant le maillage hexagonal des fibres. Les pics secondaires ressemblant aux précédents sont de l'aliasing. Le reste du plan de fourrier ressemble à du bruit blanc.
On remarque que le pic représentant la fréquence quasi verticale des fibres est très proche de 1, ce qui indique que l'on échantillonne avec une période proche de celle des fibres. Il faut donc augmenter la fréquence d'échantillonnage verticale. Si on double cette fréquence, on diminue très nettement l'aliasing (figure 13). Celui-ci reste cependant nettement visible sur le TF, mais n'apparaît pas sur l'image. Ceci permet de justifier les calculs précédents...

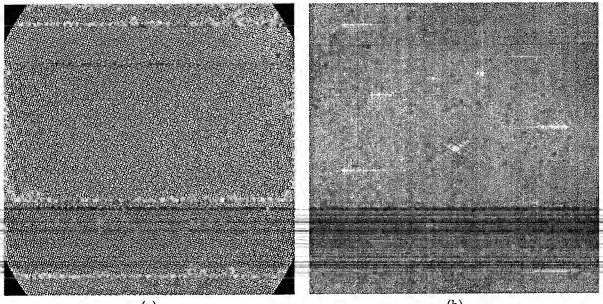
### 8.2 Technique de masquage de fréquence

### 8.3 Résultats

## 9 Modèle paramétrique de restauration

## 10 Stabilisation d'image

i.e. recalage simple.



(a) (b)

FIG. 13 – (a) Image. (b) FFT de (a).

10.1 Recalage 1D pour translation

10.2 Calibration des projections

10.3 Corrélation croisée normalisée

10.4 Résultats

## 11 Conclusion